Software-based systems – does the bar need to be raised again?

R Gandy

13 Woodkind Hey
Spital
Wirral
CH63 9JY

Keywords: criticality; governance; procurement; software

Abstract

The NHS and public sector have an unhappy history in relation to the successful introduction of computer systems. Procurement processes place responsibility for the efficacy of the software squarely with the providers, yet failures still occur. However, other sectors (eg the avionics and nuclear industries) have continued success because the criticality of their systems means that failure cannot be countenanced. This paper argues that there is much to be learned by the NHS and public sector from these other sectors, and benefits to accrue by adopting similar approaches. This would mean that improved Governmental procurement processes and governance arrangements need to be put in place that anticipate the fact that service delivery is becoming increasingly dependent on software-based technologies, where criticality is relevant.

Introduction

The NHS and other organisations in the public sector require providers of software-based systems to have standards that ensure the efficacy of their software. Of course, the providers do have standards, yet problems continue to occur which add to the unhappy history of public services, in terms of successfully introducing computer systems. This suggests that current standards are not adequate.

This paper asks the question of whether there would be benefits in the public sector adopting the rigorous approach that applies in certain other sectors where criticality is dominant.

Current approach to software validation

The current Governmental approach places the onus on providers to ensure the efficacy of software. For example, in November 2004 the Minister of State for Health, referring to the National Programme for Information Technology's Local Service Providers (LSPs)[1], stated that the procedure for accepting any software into the National Programme is extremely strict, and responsibility lies with the LSPs. Therefore it is "in their highest interests to assure themselves of the quality of software engineering" [1]; something that cannot be done collectively or by proxy.

Reference was made to major subcontractors being accredited to the Carnegie Mellon Software Institute's Capability Maturity Model, Level five (CMM5)[2]. Meeting this standard was quoted as being "very demanding", with any LSP "which did not deliver an appropriately high quality of product for National Programme software" subject to incurring "a very high

cost" [1]. This is clearly a defensible stance for any commissioning organisation. The onus for ensuring software standards is clearly placed on the provider and if the software falls short of meeting what is stated in the specification then the commissioner will place penalties on the provider (according to the severity of the shortfall).

But CMM5[2] requires an entire organisation to be focused on continual process improvement, ie it has the means to identify weaknesses and strengthen processes proactively, with the goals of preventing defects and improving efficiency. However, this is not the same as requiring affiliates to show that their software is free from defects and faults: it simply requires a reduction of defects from previous CMM5 levels to be attained. Therefore when errors or problems do arise with software they are addressed, either with a patch or an appropriate re-write. Of course, such problems will not necessarily arise during implementation itself, but may occur at a later stage when certain circumstances apply. This is only acceptable when there is little or no criticality associated with such systems and software. But as health care and other public services become increasingly dependent on systems and equipment that use software how long will this be the case? Why should the public sector not require the very highest standards and best practice, as espoused in other fields and professions?

Sectors where the criticality of software is paramount

Imagine if the same approach as that described above for the public sector was applied to the avionics and the nuclear industries? An aeroplane full of passengers is flying at 30,000 feet when the software goes wrong: what are the potential implications? It is not really practical for the pilot to send an email to the software provider to ask for something to be done as soon as possible. Criticality is central where software failure can be life critical, and relates to equipment as well as systems. Therefore, the efficacy of software standards in these sectors is paramount and the assurance and audit systems that are put in place are aimed at ensuring that software works first time and every time.

The motivations can vary between sectors: legal costs and industrial image (avionics); reliability of mechanical exchanges (telecommunications); recall costs (automobile); and avoiding disaster (nuclear). The approach applied in sectors where criticality is key requires software to be formally regulated by the main responsible organisation(s) (eg the Federal Aviation Authority in relation to avionics in the USA). Standards are agreed by consensus and applied industry-wide (eg DO-178B[3] and IEC 61508[4]). It helps if there is a long-established culture of co-operation.

Having set the standards the key question is then how to check that quality is actually in place? This requires the installation of appropriate audit and validation mechanisms. From a technical perspective, to ensure that there is no, or minimal, risk of error, the quality of any software can only be satisfactorily checked by analysing the source code; and in practice analysing the source code needs to be automated so as to cover potential permutations and combinations. To do this, software providers audit/validate their software through the application of thorough analysis tools provided by specialist companies (many of which are based in the UK). The services provided include static and dynamic analyses capabilities, the provision of tools, software toolkits, and training.

Of course, the results of the application of such audit/validation need to be appropriately acted upon. To this end, software providers in critical arenas often have a designated engineering representative (DER), or equivalent, to enforce the quality of software. This person can be an employee of the software provider, but he/she is invested with full responsibility for checking the software and ensuring that it meets the set requirements. In avionics in the USA, the DER has to sign a contract to say that they can personally be prosecuted if

standards are not met! Therefore no software can be released unless and until it has been signed off by the DER. This is essential because failure really matters where it is life critical. As a result of this type of approach high reliability can be and is achieved.

Why hasn't the NHS and the public sector adopted such standards and approaches?

Procurement processes in the public sector are usually competitive and involve a fixed price. They place the risk for software quality with the service or software provider. Arguably, this is a politically pragmatic approach: Ministers can advise Parliament that the efficacy of software has been addressed by providers being required to meet established quality standards, backed up by applying penalties where things go wrong. The drivers for public sector procurement can also include ambitious deadlines to support political programmes, where Ministers wish to see demonstrable changes and improvements in services. Service and software providers have often played catch-up as the specifications for computer systems can only really be finalised once the change/improvement programme they are meant to support have been finalised. Also, by their very nature Government systems are large and are therefore only likely to attract interest from large software providers.

Setting specification requirements such as a need to meet CMM5[2] is a straightforward option for public sector procurement, because something has to be included in respect of software quality and an approach similar to that for critical systems would require clear, identified standards to be set in the first place.

Given the above, perverse incentives can apply: providers are working within tight finance and time constraints; if stated quality requirements are satisfactorily met then is there merit in software checks to the levels involved in critical systems? – if anything were found then rewriting adds costs and time; and there could be legal liabilities/ obligations if defects are found. Ignorance can be bliss!

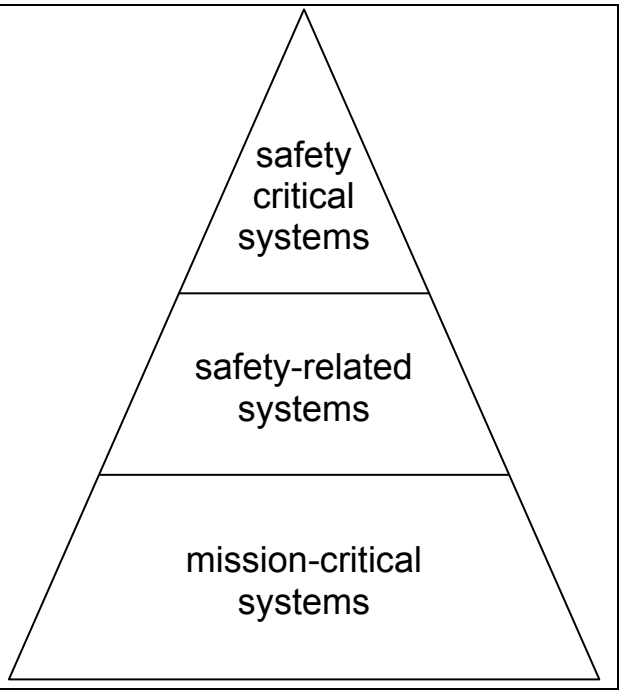
Examples	
New medical technologies (eg pacemakers) Gamma-rays for cancer Robotics in aseptic dispensing	
Air traffic control/ Ground systems Navigation systems	
Information systems CfH Social Security	

Figure 1. Criticality in NHS and public sector systems

Criticality in future systems

Criticality may not be central to many health/public sector systems at the present time, but future developments in systems and medical equipment are likely to raise its profile. Figure 1 provides an illustration of how criticality might be related to NHS/ public sector systems and equipment:

- *mission-critical systems* are those that would not endanger life if they went wrong;
- *safety-related systems* are those that do not endanger life directly if they go wring, but the consequences of their going wrong could potentially lead to this; and
- *safety-critical systems* are those that directly endanger life if they go wrong.

What might be the potential benefits of the NHS/public sector adopting such standards and approaches?

There are genuine potential benefits accruing from improving the approach to ensuring the efficacy of software:

Political benefits

- The political temperature has been rising for some time in respect to improving the efficacy of systems and the need for good governance.

Technical benefits

- Remove software faults and defects, and ensure third party software is of appropriate quality.
- Maximise the security of systems to guard against hackers and assure integrity in multi-user, sharing environments.
- Identify and address unexpected and unwanted functionality.
- Attempt to speed up developments can involve multiple teams, leading to the potential for 'code bloat' where common elements that should be developed once and shared are inadvertently duplicated.

Quality benefits

- Ensure best practice.
- Generic tools and products can be applied to practically any software from any country.
- Big reductions in risk.

Financial benefits

- A report was commissioned by the EEC to investigate the improvement of software quality through improved testing. This project is the Prevention of Errors through Experience-driven Test Efforts (PET) ESSI Project No.10438[5-7]. The PET project objectives were to reduce the number of bugs reported after release by 50%, and reduce the hours of test effort per defect found by 40%. Both of these goals were met by using available technologies. The actual numbers achieved were 75% less defects reported and a 46% improvement in test efficiency.
- Efficient processes that minimise the need to correct and re-test software are clearly cost-effective. Being able to demonstrate the application of such processes is also relevant for due diligence and can contribute to legal defence.
- The benefits of defect removal are not only financial and temporal, there is also the collateral benefit of user satisfaction through the higher dependability of the software products.

NHS-specific benefits

- Security is key.
- The NHS is subject to many pressures that will change the requirements and ground rules for IM&T. These are political, organisational, managerial, clinical and technological, and such changes can occur within a short space of time. The NHS is also one of the biggest and most complex organisations in the world. With the ongoing process of modernisation, the NHS will look very different in, say, ten years time to what it looks like now.
- The NHS track record in specifying its requirements is not necessarily a good one. As the NHS changes, so will its requirements, and it is not necessarily possible to anticipate these. Therefore LSPs will face a continually evolving specification, which could end up very different to what was originally foreseen. LSPs will endeavour to develop their software systems in line with such changes, but there is a real danger that they will become stretched and subject to unexpected and unwanted functionality. Therefore it is absolutely essential to ensure that all new software (modules) are able to fully integrate with existing systems at each and every stage, whether that software has been developed in-house or bought in.
- Criticality will become increasingly important within NHS computer systems and medical equipment, as professionals become increasingly dependent upon them to deliver patient care. In an increasingly litigious NHS, it is highly probable that at some point a patient's death or suffering could be attributed to software problems, with consequences for the company responsible.

Conclusions

There is always likely to be tension between the desire of politicians and procurers in the public sector to acquire systems and implement them as quickly as possible, and the time dictated by the logistics and demands of assurance and audit processes where the criticality of software is key. This needs to be fully recognised and taken into account as the Government moves from a position of itself being a (main) producer of software to one of having a customer focus. Equally, full account must be taken of the risk that a provider might fail in its service delivery – possibly due to problems with its software – because where this happens considerable time, effort and money is lost, and things can be back to square one. Consequently public sector procurement processes and requirements need to ensure the efficacy of the software used by providers. However, whether there is suitable intelligence within Governmental agencies around the subject area of auditing the criticality of software is uncertain.

Criticality in respect of healthcare systems is self-evident in many cases, and its importance is only likely to grow over time as service delivery becomes increasingly dependent on software-based technologies. The Government has to anticipate this and plan accordingly.

The mechanisms for an improved approach are known and therefore the question has to be asked "Why not?" Undoubtedly there will need to be a tailoring of the approach for the public sector. For example, might DERs relate to the Health and Safety Executive? The situation of where providers use commercial off-the-shelf software might present issues in terms of accessing source code. Nevertheless, the debate and decisions about how to maximise the efficacy of software-based systems for the public sector need to take place sooner rather than later.

References

1. Minister of State, Department of Health. Personal communication. 18 November 2004.
2. Carnegie Mellon University Software Engineering Institute. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, 1995.
3. European Organisation for Civil Aviation Equipment. Software considerations in Airborne Systems and Equipment Certification (ED-123/ DO-178B). Eurocae, 17 Rue Hamelin F-75783, Paris Cedex 16, France, 1992.
   Available at: www.eurocae.org
4. International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems (IEC 61508). International Electrotechnical Commission, 3 Rue de Varembe, Geneva, Switzerland, 1998.
   Available at: www.iec.ch/61508 (Accessed 17 Nov 2007).
5. Vinter O, Poulsen P-M, Nissen K, Thomsen JM. The Prevention of Errors through Experience-driven Test Efforts. ESSI Project 10438, Final Report. Brüel & Kjær A/S, 2850 Nærum, 1996.
   Available at: http://inet.uni2.dk/~vinter/finalrep.doc (Accessed 17 Nov 2007).
6. Vinter O, Poulsen P-M, Nissen K, Thomsen JM, Andersen O. The Prevention of Errors through Experience-Driven Test Efforts. DLT Report D-259. DELTA, Venlighedsvej 4, 2970 Hørsholm, 1996.
7. Vinter O. Experience-driven Process Improvement Boosts Software Quality. Proceedings of the Ninth International Software Quality Week, Software Research, San Francisco, USA, 1996.
   Available at http://inet.uni2.dk/~vinter/sqwpap96.doc (Accessed 17 Nov 2007).