Public ICT Procurement: Maximising Quality Whilst Minimising Risk

Dr Rob Gandy[1] and Professor Mike Hennell[2]

[1]Consultant & Visiting Professor, Liverpool Business School,
Liverpool John Moores University,
Redmonds Building, Brownlow Hill,
Liverpool L3 5UG, United Kingdom
E-mail: rob.gandy@ntlworld.com

[2]Technical Director, LDRA Ltd,
Portside, Monks Ferry,
Wirral CH41 5LH, United Kingdom
E-mail: hennell@ldra.com

Introduction

The UK Government published its new information and communications technology (ICT) strategy in March 2011[1]. However, it did not make a great deal of reference to the key subject of assuring the quality of software that is either procured or acquired by the Government. The authors view this issue as critical for success, and they had first highlighted the associated risks, and how they should best be addressed, to the UK Government in 2006, when they had met with senior civil servants. Their focus was the integration of the assurance of software quality into public sector procurement processes. Further meetings took place with senior representatives of the Government Procurement Service in 2012 which resulted in their being invited to submit independent "Third Party Guidance" which covered both the available procurement process options and the technical issues and opportunities (drawing on lessons from Industry). The guidance was submitted in February 2013, and after due consideration, it has now been circulated to Government teams that have an interest in this area, including the Government Procurement Service itself.

This paper summarises why the integration of the assurance of software quality into public sector procurement processes is important, and the "Third Party Guidance" submitted. The latter divides into two components: Technical aspects and Procurement processes. It is considered that the guidance is applicable to other countries as well as the UK.

Software Quality

The public sector has an unhappy history in relation to the successful introduction of computer systems, with examples in the National Health Service (NHS), Defence Infrastructure, Benefits and other departments [2]. Whilst procurement processes place responsibility for the efficacy of software squarely with the providers, failures still occur[3].

A basic premise in the purchase of any artefact is the balance between price and quality, with the general rule (assumed) that the higher the quality the higher the price. Yet the particular problem for information systems is that there is usually no discernable quality assessment visible to either the purchaser or even the provider.

There are industrial standards to ensure process quality for both hardware and software. The software component of any system is usually the most problematic, in part because there are a number of different, existing standards. These vary from the largely arbitrary and advisory (e.g. Carnegie Mellon Software Institute's Capability Maturity Model, Level Five (CMM5)[4]) to the prescriptive (e.g. Avionics' DO-178b[5] and Electro-technical's IEC61508[6])[3]. In addition, the Motor Industry Software Reliability Association (MISRA) has a range of programming standards[7]. (It is worth noting that MISRA was a UK Government-funded initiative, which has now been adopted worldwide).

CMM5 has been used in the procurement of public sector systems, but CMM5 requires an entire organisation to be focused on continual process improvement, i.e. 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. This falls short of what is required, and so greater emphasis should be placed on the level of criticality involved in systems[3].

Industries where criticality is paramount are avionics and the nuclear industries. For example, what are the potential implications for an aeroplane full of passengers flying at 30,000 feet when the software goes wrong? 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. Imagine if the same approach was applied for the public sector?

"Criticality" for public sector systems can be described in three levels:

Safety Critical: (E.g. New medical technologies (e.g. pacemakers); Gamma-rays for cancer)

Safety-related: (E.g. Air traffic control/ Ground systems; Navigation systems)

Mission-critical: (E.g. Information systems for NHS and Social Security) [3]

In addition, there is the implicit "criticality" requirement of security; it is essential that software systems do not have vulnerabilities, or other weaknesses, which will enable unauthorised access to data or other systems. These four criticalities can all be addressed by quality assessments of software.

The two examples of prescriptive standards, quoted above, manage the whole of the software life-cycle and have a very successful track record. They are both mainly directed at process quality, although they do have some product quality components. In particular they both specify objective success criteria for the testing process. This means it is known how well-tested a system is, and if this is not good enough then what must be attempted next (in relation to the relevant criticality levels).

Quality arises from three sources: people, process and product. It is arguable that when the people factor dominates the result is "Art". It is considered that the ICT industry has dragged its heels on the issue of qualifying people, and there are concerns at a lack of computer programming skills in the UK[8,9]. The situation is exacerbated by the fact that most computer professionals are poorly educated concerning quality issues. It is therefore concluded that future public sector ICT systems will carry risks relating to the quality of software, unless an appropriate approach is taken.

Taking Advantage of Current Processes

Given the above situation in respect of software quality, Government staff will have little or no idea of the (real) impact that requested changes to a procured ICT system's specification might have. Consequently, when changes are inevitably requested, software companies are able to use these as an excuse to make extra/ additional charges, and Government staff are in a weak position to argue.

Therefore companies can sometimes submit artificially low "bid" prices, in order to get the contract, knowing that they will recoup money when the inevitable specification changes arise.

Also, it is believed that Government-related work has a mark-up premium. An example of how such a premium is in operation is the fact that in the USA healthcare/ hospitals are private, and therefore independent. Hence computer companies have to be genuinely competitive with one another in respect of quality and price, in order to get business, with the inference drawn that prices are cheaper.

Quality Assurance of Software

Software product quality assessment is an area where the UK has a particularly strong commercial presence. For example, the product quality for the F35 Joint strike fighter of the USA and UK is principally assured by British manufactured tools. The same is true of the Chinese Space Programme.

The issue of how appropriate the proven techniques of such companies are to more general software is the subject of debate. That they can definitely be applied is beyond question, because commercial companies have already applied them, on their own initiative, to banking, finance, telephony, data bases and medical systems. There are no technical arguments against their use, the arguments are purely commercial.

Some commercial arguments claim that an increase in quality can be achieved only at an escalation in cost. The escalation in production cost is claimed to be prohibitive. Two factors can be considered in response: firstly that the cost of the current production process - which frequently leads to failure - is already very high, with final costs always significantly above those originally quoted or estimated; and, secondly that with modern automation techniques the escalated costs are actually marginal, and may in the end prove significantly less than the current costs.

The principal reason why commercial interests exaggerate the costs is that they do not have the requisite expertise either in their management structures or their personnel. It is also true that the small organisations which do have the expertise of working to rigorous standards do not have the expertise to build large systems.

Independent "Third Party Guidance"

The independent "Third Party Guidance" submitted by the authors to the UK Government divides into two components: Technical aspects and Procurement processes. These are set out below.

Technical aspects:

Whilst Government is pursuing an ambitious ICT strategy aimed at supporting and improving public services[1], at the same time it will seek to avoid the well-known historical problems relating to Government Acquired Software (GAS) that have occurred in the past. Such systems have not functioned according to expectations and their reliability is often inadequate. In this context reliability means the ability to perform the tasks required by the users in a timely, convenient and accurate manner.

It is strongly contended that one of the main contributory factors to the problems experienced with GAS is the efficacy and quality of the actual coding of the software. This guidance provides advice about how best to address and eliminate coding quality problems so that high standards are maintained, and new GAS systems are successful. It looks at some of the issues relevant to GAS and makes recommendations which are general in principle, practical in nature and cheap (for Government) to implement.

Perceived Causes

The causes of the problems are many and varied. In the distant past the blame was attributed to civil servants who were said to be unworldly and inexperienced, and who imposed their blinkered view on society. The answer to this offered by the ICT industry was 'leave it to people who know what they are doing'. In reality not much changed. The causes are much deeper.

An unfortunate by-product was that Government reduced the number of appropriately qualified civil servants.

Today there is the perception that avaricious ICT suppliers fool the naïve civil servants and Government ministers. This view is also too simplistic although the principle of exploiting the Government still persists.

Root causes

Most modern software systems are large and highly complex and this has actually been true for some time.

At the formulation of a new project those developing the specification rarely see the full extent of the complexity. Their view is dominated by the anticipated high level functionality of a proposed system. As requirements are drafted and expanded in greater detail it usually becomes apparent that there will need to be mechanisms to cope with faulty inputs, incorrect assumptions, inadequate responses, and so on. The processing of these bolt-on protective and corrective actions (PCA) gets steadily more complex as the system is developed. In large systems the mutual interactions of these extra processes can themselves cause massive complexity problems.

The single most influential factor in causing unwanted PCA interactions is the changing of requirements. A change in requirements rarely leads to a global review of the impact on the PCA. Unfortunately, system builders can fail to appreciate the incredible stupidity of some users of IT systems and the potential impact of their actions. An example of the latter is the China Airlines Airbus flight 140 crash in Japan where the pilot was so determined to land the aircraft that he fought the automatic go-round system which he had himself (unknowingly) selected[10]. The subsequent crash killed 264 of the 271 people aboard.

The software industry has long known that faulty or inadequate requirements are a major cause of software-based system defects. Whilst requirements analysis[11] has come a long way there is still no general means of ensuring

that there are no missing, misunderstood or unnecessary requirements which could adversely impact upon a complex system.

Faulty Assumptions

One the most common solutions or partial solutions proffered to address the problems is that using tried and trusted systems as the basis for new systems will lead to more reliability. The reason why this is too simplistic is that the perceived reliability of software systems (the users' view) is highly dependent on the patterns of use. The best example of this is are some popular commercial off-the-shelf (COTS) word processing packages which are extremely reliable when putting together a simple document, but can be extremely unreliable when handling large documents with many changes of font, colour and layout. Unfortunately the pattern of use of software is rarely measured or evaluated, and so the inclusion of tried and tested systems can be extremely risky, or at best the risk is unknown.

The current direction of Government procurement for ICT is to build around, or with, COTS systems because they are cheap and are believed to be reliable. It is this last claim, which can be untrue in specific circumstances, that is likely to lead to future problems. Without reliable measurement of patterns of use there can be no certainty and the consequences for a wrong or inappropriate assumption are entirely unpredictable.

Potential Solutions

There are potential solutions to the problems facing GAS and some of these are discussed below:

Contractual mechanisms.

This involves the use of a procurement contract that clearly and unambiguously states the obligations and responsibilities of the system supplier. This enables Government to demonstrate that the blame for system failure lies with the supplier who can then be sued. Whether suppliers will accept such a contract is problematical, the biggest problem being that the supplied system will still be faulty and fail to perform the tasks. The users of such a system are unlikely to take comfort from the apportionment of blame.

Requirements freeze.

Since many of the problems which arise are attributable to changes of requirements then there comes a point beyond which changes are not allowed. The consequence of this constraint is that the resultant system will be dated and suffer from inadequacies. Nevertheless, what it does do might be done (very) well. Systems built with this constraint are almost all in the Avionics domain. Governmental software, subject to changes in legislative alterations might be harder to treat this way.

Wrappers.

There is a school of thought which believes that unreliable systems can be made reliable by means of a wrapper system. This is some software which filters the inputs and ensures that the only permitted inputs are within the patterns of use which are regarded as reliable. The success of this approach depends on knowing the acceptable patterns of use and being able to distinguish the others. As mentioned above, there is rarely available data and making optimal decisions can be difficult.

Agile techniques.

The premise is that complex software systems can be generated quickly and easily using automation techniques. Thereby a system can be generated quickly before requirements change, and then if faults are discovered a 'fixed' system can be quickly regenerated. Similarly if new requirements do arise then again the enhanced software can be quickly generated. The software industry has a long history of 'magic bullets' which subsequently prove to be largely ineffective and there is a possibility that this is another.

So-called "Agile techniques" impose a specific structure on a software system. The ability to make changes to the software (in response to changing requirements) depends on how the current structure impacts on the new structure required to accommodate the changes. The technical term for this effect is "structure clash".

Agile techniques may make it appear simple to change a system in response to changing requirements but it may not be simple to retest the new system. Some purveyors of Agile techniques have been known to solve this issue by intimating that retesting will be unnecessary. Yet, all the above issues about

requirements completeness and PCA remain unchanged. It may be that only a small number of tests will establish the correctness of the changes or it is possible that the whole system will need to be tested from scratch; it is the internal structure which dictates the difference.

Fall-back stance

The fall-back stance in the software industries is to claim that full functional testing will or has taken place. The problem is that it is not known how to quantify or qualify functional testing, and therefore this term can be the refuge of those who wish to be unconstrained. Questions such as "How many functions are present?" and "Which have been tested?" are unanswerable. Neither in general can functions be qualified as to which are important, primary, high level, critical, etc.

In essence none of the above potential solutions is general enough to act as a complete general solution, although all have merits in particular cases. However, they can be applied on a "horses for courses" basis.

Looking for success evidence

If a search for established software quality is undertaken, three industries stand out in terms of the consistent, successful implementation of software systems.

1. Nuclear: This industry has an excellent record for producing highly reliable software based systems. They are however usually relatively small and tightly constrained by industry-wide standards.

2. Avionics: The achievements of this industry are very significant. The software systems are moderate in size and again tightly controlled by industry-wide standards. Large systems are in hand and as yet there is no evidence to suggest that success is not possible.

3. Telecommunications: The achievements of this industry are considerable. The software is not regulated by international standards but there has always been a culture of Quality. Much of the early research into software quality was performed by this industry. The industry is known to implement many software standards.

What can be distilled from these industries is that highly reliable software can be produced when the whole process is controlled by rigorous standards. The Automotive and Electrotechnical industries have noticed this and have recently introduced similar standards such as IEC 26262 (which is an adaptation of the functional safety standard IEC 61508[6] for automotive electric/electronic systems). Other industries such as Finance have been reluctant to follow, and this is arguably reflected in some of the problems witnessed in that sector[12,13].

It should be noted that all the above successful industries world-wide heavily utilise software audit and quality assessment tools. Accordingly, the application of software audit and quality assessment tools is an integral part of ensuring the efficacy, and therefore the success of the software.

There are some examples of software produced in the absence of standards which are known to be highly reliable through independent assessments. The open source UNIX operating system is one such example[14]. It was produced initially by a telecoms organisation and has subsequently been modified and extended by a host of enthusiasts. The VMS system is even more reliable, although it is now (relatively) seldom used. Both are much more reliable than some popular systems that are widely-used in the Public Sector.

By comparison, the open source GNU compiler systems which are again produced by an army of enthusiasts are more problematic and opinions vary. Open Source is not of itself a solution.

Whilst there are a host of other software systems (some free or cheap) there have been no systematic efforts to assess their performance characteristics, and this would be a suitable subject for research.

Industrial perspective

In the past GAS projects have been much sought after by the software industry. The reason is that they are usually large and highly profitable. In the past, cost escalation was an accepted practice and hence a software house could come to no harm. The problem is that moves to tighter fiscal control have always been offset by the changing of requirements, i.e. the cost escalation arises when the customer requests changes to the requirements (and because they were not part of the contract they incur extra charges).

Up to the present the odium of being associated with failing or inadequate systems has always been avoided by pointing to inadequate requirements (the customer's fault) or the changing of requirements mid-project (again the customer's fault).

A movement to a scenario where blame can be applied based on a contract is unlikely to be welcome although the previous mitigations are still likely to be applicable. It is also possible that changes to the requirements will be significantly less welcome than previously.

Way forward

A confrontational environment is rarely a recipe for success, but contracts which are clear and unambiguous in terms of tasks and responsibilities are always preferable.

There are a number of basic blocks to any software systems, e.g. operating system, data base system, communication mechanisms. In a risk-reduction scheme it almost always pays to build on the most reliable basic block rather than the cheapest. Where these quality characteristics are not known they should be explored by means of research contracts. The acceptance of assurances by the manufacturers should not be enough.

Manufacturers are reluctant to either produce quality assessments of their products or to adopt quality standards. However, the onset of security problems is changing this situation: manufacturers are under pressure (from some Governments, particularly the US) to demonstrate that their software conforms to emerging standards for security related software defects. Government should require all suppliers to state the quality standards to which their software conforms. No supplier can afford to ignore such pressures, particularly if the major Governments were to take such a stand. In Avionics it is normal practice for the suppliers of software tools to demonstrate that their products conform to quality norms. There was initial reluctance but suppliers could not ignore market pressures.

Where software is custom built there is absolutely no justification for not insisting (contractually) that the software satisfies appropriate quality standards. Claims that this will infringe intellectual property rights are unjustified. There are large sections of the software industry that already have to demonstrate quality and there is no evidence that their intellectual property rights are infringed.

One option for further strengthening the assessment that software satisfies agreed standards is by ensuring that the attainment of these standards is checked by independent quality assessors. This is standard practice in the Avionics, Automotive and Nuclear industries and has the advantage that the additional cost incurred by Government will be slight since there are many qualified and experienced assessors available.

Ultimately Government should look again at the creation of appropriate standards for software. In the past it has funded standards for the Automotive industry which have been widely accepted[7]. By comparison, the standards developed by the Government-funded Alvey project[15-17] were not accepted by the commercial software industry. It can be inferred that the reasons for the acceptance/rejection of the recommendations of these projects were the motivations of the relevant industries and had very little to do with the quality of the standards themselves. Creating appropriate standards today could be accomplished quite quickly by building on this past work and enforcing the use of the resultant standards (or any applicable standards that already exist elsewhere).

Recommendations

This Technical guidance has briefly examined many of the issues which arise in the procurement of software-based systems by Government. It has looked at many of the claims made about software and offered criticism for many of the claims.

Finally there are a number of ways in which the risks inherent in such systems can be reduced. These are:

- Award research contracts to University-led groups to explore the reliability of COTS software systems.
- Contractually demand independent quality assessments of all procured systems.
- Ensure conformance to software quality standards by the use of independent assessors.

The cost of these recommendations is tiny compared to the cost of failure or even partial failure of Governmental systems. The last two of these recommendations are merely an extension of a requirement that software producers should produce evidence that their products do not have security vulnerabilities which is likely to be a major requirement of the future.

Procurement Processes:

Whilst the Technical aspects described above are all very important, Governments need to ensure that their procurement and acquisition processes relating to ICT Systems appropriately take them into account. Putting to one side the purchase of Commercial Off The Shelf (COTS) systems, there are two main opportunities for acquiring new ICT systems:

The first is through a procurement process, where software companies are invited to put forward submissions to deliver an ICT product to meet a stated requirement. Depending on the nature of the product, each company will be expected to either present how it would develop the required solution, or present an operational solution.

The second opportunity is where a company approaches the Government will a novel ICT product, perhaps already proven in a commercial or industrial environment, which can be adopted where there is a gap in the Government's existing systems and processes.

As the Government is keen to open up the ICT market with opportunities that encourage small and medium-sized enterprises (SMEs), the question arises about how the Government can assure itself that such companies are able to deliver ICT products that are of the required standard? There is the real danger of a "Catch 22" situation where companies might be excluded from procurement processes, because they do not have the set 'track-record', yet they cannot develop the 'track-record' because they have not won any Government contracts (or similar). In principle, there could be many companies that wish to submit proposals, and whilst there are the standard financial and corporate checks that the Government might undertake to ensure that they are bone fide organisations, how might the Government check that they have the skills/ expertise to produce top quality software? It would be time-consuming and costly to the tax payer if it were concluded that the Government itself should somehow directly check the efficacy of each interested company's software. Yet without some such check, the Government will be vulnerable to companies not delivering the quality of product that they promised.

At the same time it should be recognised that ICT products continually evolve, in part to reflect the developing needs of the user organisation. It follows that the Government should ensure that any future specifications should be structured and written in such a way that there is a "core" specification that should not change over time, with scope for flexibility so that additional components/ requirement can be added later, if the need transpires. This would probably require there to be a suitable external interface. Hence the structuring of requirements in a specification is a critical issue.

Way Forward

To ensure that the quality of software is maximised to support the Government's ICT strategy[1], minimise the risks associated with poor-quality software, and open up the market, it is proposed that interested companies should be required to demonstrate that they meet whatever is set as the requisite standard, in the full knowledge that if they are unable to do this they will not be considered by the Government. How might this be done?

There are a number of options, based on the fact that the quality assurance and audit sector of the international ICT industry already performs this type of function for a wide range of software applications, including aviation and the nuclear industry. This involves relevant software quality assurance/ audit companies checking the efficacy of systems to ensure that the required standards are met. Existing standards usually relate to safety critical systems and safety-related systems, (e.g. DO-178b[5] and IEC61508[6]), and systems with security implications [18], but there is no reason why the Government should not develop or set out equivalent standards for mission-critical systems[3]. Also, some programming standards, such as those from MISRA[7], can be applied irrespective of software criticality.

In all procurements it is assumed that there would be a clear statement as to what minimum level of software standards needs to be met, with a view formed on an ad hoc basis where a company makes a direct approach.

The options are set out below:

> Option A: Accreditation Approach
>
> The Government should accredit established, reputable software quality assurance/ audit companies to be able to independently check companies' software and certificate that set standards have been achieved. This would clearly be appropriate in respect of any existing software that is proposed for Government usage.

Where the Government is seeking to procure the development of a system then companies would be required to show that they can develop such systems. For this, the Government would require that companies achieve appropriate certification for a software system that the company can demonstrate has relevant equivalence to that to be developed.

In this way there is a clear gateway for companies to enter the Government ICT market, and the cost of achieving certification is borne by the prospective companies rather than the Government.

Related course of action:

1. Government sits down with representatives of the quality assurance and audit sector of the international ICT industry to agree classification of software standards that might be applicable for the full range of anticipated systems, together with what might be the minimum standards for each level.

2. Processes are agreed for how accredited software quality assurance/ audit companies should certificate software to meet Government's requirements. (For example, certification may only be deemed valid for a set period).

3. How such certification should be integrated with the Government's ICT procurement processes needs to be clarified, and the necessary action taken. (For example, appropriate time may need to be built in for companies to seek and acquire certification).

4. The above should enable the approach to be 'fleshed-out', so that the mainstream ICT industry can be consulted. This might benefit from a pilot exercise, where a software quality assurance/ audit company tests an agreed, existing piece of software. A view may be formed as to the order of magnitude of fees for certification testing.

5. Thought will need to be given to any cascade/ subcontracting implications, such as where a company provides a product to Government, which itself uses software (which may or may not be sourced from another company). For example, transport and armaments use software for a range of purposes – is the Government confident of the software quality?

6. Assuming a positive outcome from the consultation, the approach is then implemented. This will include: a process to accredit software quality assurance/ audit companies to certificate software; with suitable publicity to enable ICT companies to take appropriate action.

The above course of action should not require a great deal of time, and could be completed within a matter of months. By setting explicit, transparent software quality standards, the Government will send out a clear message about the importance of software quality assurance. An independent certification process should enable the opening up of the ICT market to SMEs, at minimal cost to the taxpayer, and bring confidence to Government about proposals/ products received.

Option B: Consortia Approach

An alternative approach is to use, or encourage, consortia with three components to develop ICT systems:

- Large companies to supply the vision and overall control;

- Small companies to supply the technical expertise; and

- An independent quality assurance specialist company.

It is recommended here that it is the quality assurance specialist company which has overall project control so that all quality issues can be addressed in a timely fashion. This recommendation arises from the observation that DO-178b[5] requires persons (Designated Engineering Representatives) who have authority over all other project personnel and who sign-off that project. It is these people who ensure overall other considerations that quality is never compromised[3]. It should be noted that Designated Engineering Representatives are more powerful and more stringent than Independent Safety Auditors that relate to software [19].

Option C: Contract Requirement

This is something of a pragmatic hybrid of the above. It would have procurement processes undertaken in a similar way to that at present, but the key development would be to include a contract requirement that the successful company must arrange for its software to be quality assured by an independent specialist company, which will be responsible for signing-off the project. This may or may not specify the involvement of a Designated Engineering Representative, according to the nature of the system being procured. The cost of the independent quality assurance would need to be built in to the bid; thereby incentivising the bidders to ensure high quality software so that such costs are minimised.

In this way the Government is ensuring access to appropriate independent specialist skills at the right time, with no budgeted cost incurred on its own part. The related costs will have been built into the bids and the sums involved and the attitude of the companies would have been evaluated as part of the procurement process.

In circumstances where a company approaches the Government will a novel ICT product then involving an independent specialist company to audit/ validate the software on offer represents good sense. The cost of such audit/ validation would be the subject of debate and negotiation.

Option C is arguably the most practical way forward. Option A may be considered to be a form of "red tape", which Governments and Industries are always keen to reduce. Option B could be considered to be somewhat idealist – while collaborative consortia have advantages, there can be no compulsion in their creation and therefore they are unlikely to occur (across the board).

To support such a procurement process one possibility is for the Government agency to appoint the independent specialist company/ Auditor at the outset, to help draft the quality requirements and standards. The Auditor may also participate in the assessment of the bids. The chosen contractor must then accept the role of this Auditor and undertake to satisfy the quality requirements as assessed by that Auditor.

Discussion

The whole issue of public sector ICT systems is challenging, and includes many conundrums, such as the fact that over time a system will become more stable, but at the same time it will become more obsolete.

This analysis of potential procurement processes focuses on a particular issue that is key to underpinning the success of the whole of the Government's ICT strategy[1]. It needs to be positively addressed, otherwise there is the danger that the good that is in the new strategy will not be realised.

Making software quality and audit a contractual requirement (Option C) is a sensible and practical way forward, which can be quickly acted upon. The results from such audits should be collected and collated on an ongoing basis, so that the Government can strengthen its intelligence in this matter.

The concept of using open-source software which is a central feature of the ICT strategy[1] has much to commend it, but it has to be recognised that the quality of such software is usually completely unknown and can be considered suspect. However, whilst the process-related quality will always remain unknown the product-related quality can definitely be measured and even improved at relatively little cost. The relevant techniques are again essentially those product quality techniques already mentioned and exploited by established companies. This analysis does not challenge the policy of using open-source software in any way. It is aimed at complementing it and strengthening the ways in which (other) systems are procured, in a practical and cost-effective manner, so that those that Government acquires will be of top quality.

Conclusions

The "Third Party Guidance" described above was submitted to the UK Government in February 2013, where it has been circulated to Government teams that have an interest in this area, including the Government Procurement Service itself. Given the timeframes involved in public sector procurements it will be some time before any impact can be measured and evaluated. Nevertheless, the importance of assuring the quality software is something that all Governments need to address in their procurement of ICT systems, particularly given the current systems security agenda. Therefore the "Third Party Guidance" has worldwide relevance.

References

[1] Cabinet Office (2011): Government ICT Strategy,
http://www.cabinetoffice.gov.uk/sites/default/files/resources/uk-government-government-ict-strategy_0.pdf
(Accessed 20th January 2014)

[2] M. Ballard (2012): Project failures: MPs get in a funk over open source, ComputerWeekly.com http://www.computerweekly.com/blogs/public-sector/project-failures/ (Accessed 20th January 2014)

[3] R. Gandy (2008): Software-based Systems – Does the bar need to be raised again?, Healthcare Computing Conference (HC2008), 21st – 23rd April, Harrogate

[4] Carnegie Mellon University/ Software Engineering Institute. (1995): The Capability Maturity Model: Guidelines for Improving the Software Process, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, Ma.

[5] European Organisation for Civil Aviation Equipment (1992): Software considerations in Airborne Systems and Equipment Certification (ED-123/ DO-178B). Eurocae, 17 Rue Hamelin F-75783, Paris Cedex 16, France. Available at http://www.eurocae.net (Accessed 20th January 2014)

[6] International Electrotechnical Commission (1998): Safety of Electrical/ Electronic/ Programmable Electronic Safety-related Systems (IEC 61508). International Electrotechnical Commission, 3 Rue de Varembe, Geneva, Switzerland. Available at www.iec.ch/61508 (Accessed 20th January 2014)

[7] The Motor Industry Software Reliability Association (2014): Available at www.misra.org.uk (Accessed 20th January 2014)

[8] S. Dinneen (2011): Sega exec warns of skills shortage, City A.M. , 2nd June. Available at http://www.cityam.com/news-and-analysis/sega-exec-warns-skills-shortage (Accessed 20th January 2014)

[9] British Computer Society (2011): Consultation response to National Curriculum Review – Call for Evidence. Available at http://www.computingatschool.org.uk/data/uploads/CurricReviewResponse.pdf (Accessed 20th January 2014)

[10] Hamatura Institute for the Advancement of Technology: Failure Knowledge Database (2013): China Airlines Airbus A300-600R (Flight 140) Misses Landing and Goes up in Flame at Nagoya Airport http://www.sozogaku.com/fkd/en/cfen/CA1000621.html (Accessed 20th January 2014)

[11] L.A. Maciaszek (2005): Requirements Analysis and System Design, 2 ed., Addison Wesley, Harlow England, Pp 504 (ISBN 0321204646)

[12] R. Allison (2003): ATM gives out free cash and lands family in court. The Guardian, Thursday 16 January http://www.guardian.co.uk/uk/2003/jan/16/rebeccaallison2 (Accessed 20th January 2014)

[13] S. Foley (2012): The disasters that show why we have to be on our guard with algos. The Independent, 4th August http://www.independent.co.uk/news/business/comment/stephen-foley-the-disasters-that-show-why-we-have-to-be-on-our-guard-with-algos-8005717.html (Accessed 20th January 2014)

[14] Computer Hope (2013): Unix, Linux and variant history http://www.computerhope.com/history/unix.htm (Accessed 20th January 2014)

[15] Empiricom Solution Management Technology – Our Research Heritage (2013): http://www.empiricom.co.uk/empiricom-research-heritage.htm (Accessed 20th January 2014)

[16] E. de Robien (1990): Alvey: What was it all about? Physics World. December Pp 54-56 http://physicsworldarchive.iop.org/index.cfm?action=summary&doc=3%2F12%2Fphwv3i12a34%40pwa-xml&qt= (Accessed 20th January 2014)

[17] B. Oakley and K. Owen (1990): Alvey: Britain's Strategic Computing Initiative. MIT Press. ISBN 0-262-15038-7

[18] International Organization for Standardization/ International Electrotechnical Commission (2012) Information Technology - Programming languages, their environments and system software interfaces - C Secure Coding Rules http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1624.pdf (Accessed 20th January 2014)

[19] K. Harrison and J. Dawson (2003) Effective Independent Safety Assessment, Praxis Critical Systems Limited http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.199.3418&rep=rep1&type=pdf (Accessed 20th January 2014)