UK COMPUTING RESEARCH COMMITTEE

Voter Registration

Comments to the House of Commons Office of the Deputy Prime Minister Select Committee

Executive Summary

1. The UK Computing Research Committee (UKCRC), an Expert Panel of the British Computer Society, the Institution of Electrical Engineers and the Council of Professors and Heads of Computing, was formed in November 2000 as a policy committee for computing research in the UK. Its members are leading computing researchers from UK academia and industry.
2. UKCRC has the expertise to address issue (d) identified by the committee (‘Advantages or disadvantages of electronic rather than paper-based registration systems’), and the related issue (i), at least as it concerns the security of computer-based systems.
3. Since we have no details about the scope of a proposed voter registration system, many of our remarks here apply to computer-based systems in general.
4. We begin with a brief discussion of the advantages to be gained, in principle, from the use of computer-based systems in some applications. Examples of these advantages include the flexibility that comes with programmability, and the possibility for greater and more complex functionality than is possible in a paper-based system.
5. There follows some discussion of the problems that can arise in the use of these systems. Many of these concern dependability – ensuring that users can justifiably place trust in the system’s operation.
6. We explain why these problems are difficult, e.g. *how novelty, difficulty* and *complexity* can undermine trust in dependability.
7. Finally, we discuss at some length the problems of *project* risk: why it is that projects to build new computer-based systems often fail. In the worst cases, many projects have been abandoned without producing a working system. Even when a system is delivered, it often falls far short of its users’ expectations. Project cost overruns are commonplace, and often massive.
8. At this stage of incomplete definition of the required voter registration system, any assessment of risk can only be generic. Decisions about procuring such a system need to include a more precise definition of the required system, its uses and responsibilities, a hazard analysis based on it, and thus a specification of the safeguards to be used in designing and developing the system and those that must be included in the system itself. Only on this basis will it be possible for both the government and contractors to evaluate the cost and feasibility of such a project.
9. UKCRC would be happy to answer follow-up questions on any of these points. It would also welcome further more detailed consultation at suitable points in the progress of the project if it goes ahead.

1. Without knowing more about the potential scope of the voter registration systems that are proposed, it is difficult to be very specific about advantages and disadvantages of electronic schemes. Many of the remarks here, therefore, will apply to computerbased systems in general.
2. The major advantage of computers lies in their *general purpose* nature, derived from their programmability. This allows much of the specific functionality in a particular application to be implemented in software and thus avoid some of the unreliability due to failures of, say, human procedures in a paper-based process.
3. Furthermore, it is possible to implement functionality in software-based systems that would be inconceivable in other technologies. In voter registration, for example, it might be easy to implement much more extensive checks against cheating (e.g. an individual registering in multiple constituencies) than would be possible to conduct manually in a paper-based system. Similarly, the range of security checks that could be applied by software-based systems may be richer than those that are feasible in a conventional system.
4. The use of software can introduce great flexibility because of the possibility of reprogrammability. Enhancement of programs (to meet changed needs, improved technology, or just better understanding) can be much cheaper and easier than in other branches of engineering.
5. Unlike humans, software is 'tireless' in the sense that if it can do something right, it generally *will* do it right.
6. Some of these advantages of computer-based systems do not come without their price, however. For example, the flexibility of reprogrammability can also be disastrously expensive, e.g. by the introduction of novel faults into correctly functioning programs. The control of such in-service change needs consideration at an early planning stage.
7. The process of eliciting the requirements of a system, and then analysing these requirements for consistency, etc, is of supreme importance. Changes to a system design at a late stage, to overcome some deficiency in the requirements specification, can be enormously expensive, as well as being a source of novel faults.
8. It is possible to implement *too much* functionality in the form of 'bells and whistles', resulting in unjustifiable complexity, and consequent unreliability (and/or insecurity) of the resulting system. It is common for designers to suffer from a kind of hubris – 'we can implement anything in software' – and take on tasks of overwhelming difficulty. Sometimes these problems are accentuated by the design being for a completely novel purpose, so that the designers have no opportunity to learn from experience with earlier systems.
9. These related issues of *novelty, difficulty* and *complexity* are important reasons why some software projects result in systems that are not sufficiently dependable,[1] and why some projects are not even completed[2]. Of course, one should not be defeatist here. It is possible, with good design, to guard against these malign influences. For example, when a computer system is being designed to replace some other technology, such as the paper-based voter registration system, it would be sensible to consider using some of the tried and tested ideas from the old system in the design of the new. In cases like this, careful design can ensure that key functionality is delivered reliably by learning from past mistakes, whilst at the same time safely incorporating some new functionality that exploits the newer technology.
10. Computer-based systems are often much more centralised than the systems they replace. In replacing a paper-based voter registration scheme with a computerised one, there might be a temptation to organise this at a national level, or at least to have extensive networked communications between regional systems. Clearly, this tendency for centralisation has its advantages – e.g. in checking for fraud in voter registration schemes. However, a disadvantage is that any accidental failures, or deliberate subversions of the system, are likely to have much more extensive consequences than they would have in a paper-based system. For instance, false registrations could be created, or legitimate ones invalidated, in large numbers, sufficient to covertly steal an election or openly make it fail.
11. In some *non-computerised* systems and processes, humans have evolved sophisticated ways of noticing deviations from acceptable behaviour. When computers are introduced into these situations, it is easy for the designer of the computer system to unwittingly eliminate these possibilities for human corrections, as these are often not explicitly recognised. At the very least, designers can create an extra burden for themselves, as they need to anticipate all possible glitches, and find solutions to them.
12. It is worth saying that computer-based systems are not a panacea for solving problems with current systems. For example, there are some inherent conflicts involved in voting systems – e.g. between privacy and auditability of recording[3]. It is important that decisions are taken about how to balance

[1] 'Dependability' is a jargon word in computing. Informally, a system is dependable if the user can justifiably depend upon the service(s) it provides. Thus attributes of dependability will include reliability, safety, availability when called upon to provide a service, resilience against malicious attacks, protection of information from disclosure to unauthorised people, and so on.
[2] For a fuller discussion of these issues, see 'The use of computers in safety-critical applications,' HSE Books, 1998. Although this report concerns safety-critical computing, many of the issues are generic and apply to the subject of this note.

these before the specifications for technical systems are written, otherwise there is a danger that naive faith in technology will exacerbate the problems.

13. Given the importance to democracy and good government of *justified* public faith in systems such as these, it is essential that the principles and procedures involved in them should be inspected and authorised by technically qualified and independent parties. In particular, commercial confidentiality claims should not be allowed to inhibit this.

14. An issue that is often given too little thought and effort is that of system assurance. Every system needs to be sufficiently dependable (reliable, secure). Often too little attention is given to what is meant by 'sufficient': all computer systems will contain faults, so a goal of complete perfection is infeasible. Instead, it is necessary to decide what levels of dependability are acceptable, taking account of the fact that, all things being equal, obtaining higher dependability will involve higher costs. The process of designing and building the system should then be informed by the required level of dependability. In particular, thought needs to be given to means of assessing dependability (both during the design process – is the system on track? – and of the finished system – is the delivered system sufficiently dependable?)

15. There are many examples of systems for which these questions of assessment have been considered too late. One should never be in the position of deciding only after a system is built how it is to be assessed. Nor should the cost of assessment be underestimated; in fact these costs can be even greater than those needed to build the system in certain cases[4]. These costs should never be seen as money wasted, but rather as a necessary reassurance that the system can be depended upon[5]. This issue of public confidence is likely to be an important one for an electronic voter registration scheme.

16. There are many techniques under the broad heading of 'assessment', but some seem particularly relevant to voter registration systems. For example, *hazard analysis* would be required to identify the kinds of possible disruptive events, and the likely extent of their consequences. Such an analysis would be necessary to inform decisions about mitigating procedures, which could be part of the computer-based system (e.g. physical redundancy of hardware) or be human-based fall-back procedures. Mitigation measures can also consist in enforcing explicit limits to the possible uses of the new system and the data it generates: an automated system may for instance make possible new forms of misuse or novel, attractive uses; but since these uses were not possible before, their possible risks cannot be learned from previous experience, and must be found by explicit analysis.

17. Our remarks so far have mainly concerned *systems*, and risks associated with their failure in operation. If the government is to embark on an extensive investment in electronic voter registration, then it is also important to consider *project* risks. By this we mean the potentiality for failure in the whole process of procuring computer-based systems: such failures include non-completion of projects, huge cost overruns, etc.

18. A recent study of problems of large-scale projects by the Royal Academy of Engineering[6] reported that poor project definition is one of the major contributors to project failure. The Academy also observed that there is a greater tendency for poor project definition in the public sector, where systems are intended to meet political ends, and the practicalities often have not been thought through.

19. Recent work by the Office of Government Commerce has addressed the management failures that have contributed to failures of Government IT projects, with recommendations relating to the appointment of 'senior responsible owners' and guidelines for Gateway Reviews. More recently, the OGC and Intellect (the trade association for companies in the IT industry) have collaborated on a Code of Conduct to address some of the worst examples of unprofessional behaviour by suppliers to Government. These are worthwhile initiatives and we support them, but in our opinion they do not get to the heart of the problem.

20. Developing new computer software is an engineering task—often of extraordinary complexity. The software for a modern air-traffic control centre, or to support a Government Agency, will involve a million lines of computer program (often, several million lines) and will typically require tens or hundreds of person-years of effort to design. It is beyond the power of humans to create structures of this size and complexity without making mistakes, so the focus of software engineering has to be on reducing the number of errors made, finding as many as possible before the software is delivered, and designing programs, and the human procedures surrounding them, so as to

[3] For further discussion of such issues see, for example, Rebecca Mercuri, 'A better ballot box?', IEEE Spectrum, vol 39, no 10, October 2002, or her website http://www.notablesoftware.com/evote.html.

[4] For example, the French RER railway in Paris contains a safety-critical system of modest size (about 20,000 lines of code); 100 person years was spent solely on assurance. The statistical testing of the software in the protection system of the Sizewell nuclear reactor cost several million pounds.

[5] Large amounts of money were spent on the 'millennium bug' problem. Some people argued that in those cases where no significant problems were found, this money was wasted. This is a mistaken view: there was genuine uncertainty here, and the money was justifiably spent to gain confidence in the face of this uncertainty.

[6] 'The Challenges of Complex IT Projects,' A report of a working group from the Royal Academy of Engineering and the British Computer Society, April 2003.

be resilient to the effects of the faults that remain, recognising and reacting to any incipient inappropriate behaviour of the software. Unfortunately (and, as we shall see, unnecessarily) most software developers deliver systems that still contain more than ten faults in each thousand lines of program. A large system will therefore contain upward of ten thousand faults, each waiting to cause a failure under the particular circumstances that cause that path in the software to be executed.

21. Engineering is much more than applied science, although the science is essential; engineers also use mathematics, codified experience, and judgement. They use mathematics to build models of their designs and to explore the properties of these designs before investing in further development. They capture experience of design methods and management in a form that can be taught as best practice engineering processes. They codify this experience in the form of standards, such as the international quality management standard, ISO9001. The best software engineers recognise the parallels between software design and classical engineering, and adopt similarly rigorous methods. Unfortunately, most suppliers do not follow these principles, and most Government Departments are not knowledgeable enough to hold them to account for their failure to reach levels of professional competence and responsibility that are taken for granted in other branches of engineering.
22. Two surveys illustrate the scale of the problem. In 1995, The Standish Group[7] reported that the average US software project overran its budgeted time by 190%, its budgeted costs by 222%, and delivered only 60% of the planned functionality. Only 16% of projects were delivered at the estimated time and cost, and 31% of projects were cancelled before delivery. Later Standish Group surveys show an improving trend, but success rates are still low. The Standish Group's 'Chaos Chronicles' report for 2003 analyzed over 13,000 IT projects, and estimated that nearly 70% either failed completely or were 'challenged' (i.e. although completed and operational, exceeded their budget and time estimates, and had less functionality than originally specified). This led to their estimate that in 2002 the US 'wasted $55 billion in cancelled and over-run IT projects, compared with a total IT spend of $255 billion.'
23. The picture is fortunately not one of unrelieved gloom. Across Europe, there are a few companies who deliver software projects significantly more quickly and more cheaply by using 'formal methods'—software engineering methods that are soundly based in computer science—and mature engineering processes. These companies are increasingly willing to offer warranties, because they know that they can deliver software with 0.1 to 1 fault per thousand lines, instead of the 10-20 faults per thousand lines that are more typical. Such companies are able to analyse the requirements that they have been given by their customers, to reveal the inconsistencies and to draw out the deeper requirements that would otherwise result in late changes to the project requirements.
24. Finally, it has to be said that not all the problems revealed by these surveys are the responsibility of the developers: some are the fault of customers themselves. For example, late changes to requirements are very damaging. The customer has to renegotiate the project cost and timescales, usually at a point where competitive bids cannot be obtained. The project risk is transferred back to the Government Department, and the project slips and runs over budget.
25. Of course some change is inevitable, but it is important that it is controlled. Here again, the use of rigorous formalisms can be helpful in ensuring that (some reasonably foreseeable) late changes can be identified rapidly, and can be accommodated by rescheduling.

26. Conclusion. At this stage of incomplete definition of the required voter registration system, any assessment of risk can only be generic. Decisions about procuring such a system need to include a more precise definition of the required system, its uses and responsibilities, a hazard analysis based on it, and thus a specification of the safeguards to be used in designing and developing the system and those that must be included in the system itself. Only on this basis will it be possible for both the government and contractors to evaluate the cost and feasibility of such a project.

27. UKCRC would be happy to answer follow-up questions on any of these points. It would also welcome further more detailed consultation at suitable points in the progress of the project if it goes ahead.

Bev Littlewood
for UKCRC, January 2005.

[7] http://www.standishgroup.com