The Challenges of Complex IT Projects

The report of a working group from The Royal Academy of Engineering and The British Computer Society

The Challenges of Complex IT Projects

ISBN 1-903496-15-2

April 2004
Published by
The Royal Academy of Engineering
29 Great Peter Street, Westminster, London, SW1P 3LW

Telephone 0202 7227 0500 Facsimile 020 7233 0054
www.raeng.org.uk

The Royal Academy of Engineering is a Registered Charity (No. 293074)

Table of Contents

Notes:

1. It cannot be guaranteed that following the guidance in this report will prevent project failure. However, there is a strong probability that ignoring it will prevent success.

2. As the study was conducted under the Chatham House rule, quotations have only been attributed where special permission has been obtained from the originator. The views expressed in this report cannot be considered representative of any particular contributor, or the organisation to which they belong

Executive Summary

"A significant percentage of IT project failures, perhaps most, could have been avoided using techniques we already know how to apply. For shame, we can do better than this." (L. Hatton)

Background

It is estimated that expenditure on IT in the UK public sector alone will top £12.4 billion[1] in 2003/04, with the overall UK spend on IT projected to be a monumental £22.6 billion[2]. Against this background, it is alarming that significant numbers of complex software and IT projects still fail to deliver key benefits on time and to target cost and specification. Whilst complex IT project success rates may be improving, the challenges associated with such projects are also increasing rapidly. These are fuelled in large part by the exponential growth in the capability of hardware and communications technology, and the corresponding inflation in people's expectations and ambition.

Study Overview

This study, conducted by a group of Fellows of The Royal Academy of Engineering and British Computer Society (BCS), seeks to improve the understanding of how complex IT projects differ from other engineering projects, with a view to identifying ways to augment the successful delivery of IT projects. The findings and recommendations are derived from the extensive body of evidence collected, in both written and oral form, from more than 70 individuals, encompassing senior directors, managers, project managers and software engineers from the public and private sector, as well as academic experts.

Findings and Recommendations

A striking proportion of project difficulties stem from people in both customer and supplier organisations failing to implement known best practice. This can be ascribed to the general absence of collective professionalism in the IT industry, as well as inadequacies in the education and training of customer and supplier staff at all levels. Moreover, there is a broad reluctance to accept that complex IT projects have many similarities with major engineering projects and would benefit from greater application of well established engineering and project management procedures. For example, the importance of risk management is poorly understood and the significance of systems architecture is not appreciated.

Whilst the most pressing problems relate to the people and processes involved in complex IT projects, further developments in methods and tools to support the design and delivery of such projects could also help to raise success rates. In particular, basic research into complexity is required to facilitate more effective management of the increasingly complex IT projects being undertaken.

There is much that individual organisations and project teams can do to improve their chances of success in IT project delivery and guidance is provided in Part II of this report. However, a number of the key findings can only be addressed effectively through concerted action at a national level. The study concluded that:

1. The levels of professionalism observed in software engineering are generally lower than those in other branches of engineering, although there are exceptions.
   (a) Customers should therefore ensure that all senior IT practitioners involved in the design and delivery of 'high consequence' systems have attained Chartered status and maintain their technical currency through Continuing Professional Development (CPD);
   (b) The Office of Government Commerce, together with the Professional Institutions, should assess means of enforcing the registration, and maintenance of professional competence through CPD, of senior practitioners working on high consequence systems;
   (c) The Professional Institutions should work together with the Confederation of British Industry and the Institute of Directors to promote awareness of the benefits of employing Chartered IT practitioners and suppliers who have adopted the Intellect Code of Best Practice.

[1] Kable Ltd, www.kablenet.com
[2] Ovum Holway Report – Holway@Ovum: Industry trends, 2002

2. Education in many universities and management schools in the UK is not producing IT practitioners with the IT application and project skills they need.
   (a) Intellect[3] should therefore take a lead in forming links between companies, government departments and universities to promote a greater applications focus in undergraduate courses;
   (b) The BCS and Institution of Electrical Engineers should produce a model syllabus or other criteria to apply in assessing and accrediting undergraduate courses, to encourage a move towards courses with a stronger engineering emphasis.

3. The importance of project management is not well understood and usually under-rated and senior managers are often ill qualified to handle issues relating to complex IT projects.
   (a) Management schools should therefore collaborate with computer science and engineering departments, as well as the project management Professional Bodies, to develop courses specifically addressing IT project management;
   (b) Management schools should also ensure that both project management and IT are core modules of MBA courses.

4. Risk management is critical to success in complex projects but is seldom applied effectively in the case of IT and software.
   The Royal Academy of Engineering should therefore produce guidance to address risks of IT projects in such a form that it can be used for corporate governance in the framework provided by the Turnbull Report.

5. The vital role of the systems architect in major IT projects is frequently not appreciated and there is a shortage of appropriately skilled individuals.
   The Professional Bodies should therefore work together with the Engineering and Physical Sciences Research Council (EPSRC) and the Department for Education and Skills to explore ways to identify and develop the skills of people with the potential to become systems architects.

6. There is an urgent need to promote the adoption of best practice amongst IT practitioners and their customers.
   Government, with the Department of Trade and Industry (DTI) taking the lead, should therefore work jointly with Industry to establish a UK Software Engineering Institute for research, advice and training to promote best practice in software engineering and IT project management.

7. Basic research into complexity and associated issues is required to enable the effective development of complex, globally distributed systems.
   The DTI and EPSRC should therefore establish a UK research programme on complex IT systems to address the design, development, evolution and assessment of complex, distributed IT systems.

The increasing prevalence of IT systems, coupled with overseas competition in this area, means that failure to improve the collective professionalism of the IT industry and strengthen the national infrastructure supporting project delivery is likely to have serious and ongoing economic consequences for the UK.

[3] Trade Association for the Information Technology, Telecommunications and Electronics Industries in the UK

Introduction

Complex IT systems are integral to the functioning of our society. They contribute to the design, production and delivery of innumerable products and services that we encounter as we live, learn, work and play, and their significance will inevitably increase in coming years. Yet horror stories of colossal IT project failures hit the headlines on what seems like a daily basis, as illustrated by the news cuttings on the inside cover. This is partly responsible for the perception, which is to some extent supported by evidence, that the success rates of software and IT systems projects are disappointingly low.

This study seeks to understand what makes complex software and IT projects different from, and potentially more difficult than, other typical engineering projects, with a view to improving success rates. The report represents the findings of a team of Fellows of The Royal Academy of Engineering and the British Computer Society (BCS), based on evidence gathered both orally and in the form of written submissions. In all, more than 70 individuals with a wide variety of backgrounds, expertise and experience contributed evidence. The names of contributors are listed in Appendix II and many direct quotations from the evidence received have been used to illustrate the points made in the report.

The report is divided into three main sections. Part I addresses possible reasons for the low success rates of IT projects through a comparison of the principles and practice of software engineering and IT projects with those in other branches of engineering. Part II provides guidance on best practice for people involved in commissioning, designing and managing IT projects. Finally, Part III summarises the high-level findings and makes a number of recommendations, with the intention of improving the national support infrastructure and thus the environment in which IT projects are carried out. Collectively, it is hoped that these will contribute to enhancing the national capability to deliver successful complex IT projects.

Box 1: Definition of a complex software or IT project

This study is concerned with large scale IT projects with a significant and complex software component. For simplicity, the term ‘IT system’ or ‘IT project’ will be used throughout the report to refer to these complex software-based systems and projects.

A number of other studies make a distinction between ‘failed’ and ‘troubled’ or ‘challenged’ projects and define boundaries for the various categories. This report uses the term ‘failure’ generically for projects which fail to deliver all the key features and benefits to time, target cost and specification.

Cause for concern

Although significant numbers of IT projects are routinely completed successfully (see case study 2), it must be conceded that the data on IT project success rates provide cause for concern. For example, a recent study on the state of IT project management in the UK carried out by Oxford University and Computer Weekly reported that a mere

16% of IT projects were considered successful[4]. Similarly, in another UK survey published by the BCS, only three out of the more than 500 development projects assessed met the survey's criteria for success[5]. Nevertheless, the Standish Group estimates current success rates in the US at around 34%, which represents a significant improvement on the 16% success rate recorded in their first survey in 1995[6]. The Oxford University/Computer Weekly survey also suggests that UK success rates, although low, may be improving.

It is difficult to quantify the financial cost arising from these low success rates, however a recent review estimated that a phenomenal US$150 billion per annum was attributable to wastage arising from IT project failures in the Unites States, with a further US$140 billion in the European Union[7]. Irrespective of the precise amount, it is clear that the price tag associated with IT project failures is unacceptably high. In addition, the increasing prevalence and complexity of software in safety critical systems, business critical systems and medical devices means that there is growing potential for high human costs arising from IT system failure.

Box 2: Public Sector IT Projects

The Oxford University/Computer Weekly survey of IT projects found little difference in the performance of the public and private sector. However, there have been a number of high profile public sector failures that have significantly tarnished government's reputation in the delivery of IT projects. For example, in 2003, the Chairman of the Public Accounts Committee described 'Libra', a new IT system for the Magistrates Courts, as "*one of the worst IT projects I have ever seen*"[8] (see case study 1).

It is generally agreed that public sector projects face specific difficulties, including their high visibility, the risk averse culture of the civil service, the need to meet politically-driven timescales and, in many cases, their enormous scale and complexity. The Office of Government Commerce has played an active role in addressing some of these issues, with a view to improving the capability of the public sector to deliver IT projects (see box 4).

Source: Government IT Projects, Parliamentary Office of Science and Technology, July 2003

[4] *The State of IT Project Management in the UK*, Chris Sauer and Christine Cuthbertson, Templeton College, Oxford, November 2003

[5] *IT Projects Sink or Swim*, Andrew Taylor, British Computer Society Review 2001

[6] *Latest Standish Group CHAOS Report Shows Project Success Rates Have Improved by 50%*, Press Release, Standish Group, March 2003

[7] *Avoiding IS/IT Implementation Failure*, Darren Dalcher and Audley Genus, Technology Analysis and Strategic Management, TASM, Vol. 15, No. 4, pp. 403-407, December 2003

[8] *Courts Libra system 'is one of the worst IT projects ever seen'*, Computer Weekly, 30 January 2003

Case study 1: Libra IT system for magistrates' courts

Libra was designed to provide a standard IT system for magistrates' courts including upgraded infrastructure, office automation facilities, a national casework application and electronic links with other criminal justice agencies. The original contract for £184m was awarded in 1998 but following implementation problems the deal collapsed in July 2002. After renegotiation later in 2002, the infrastructure portion alone cost £232m. The total system would now take 8.5 years to develop and cost over £318m, as well as being available (to deliver benefit) for two years less than anticipated.

Source: Prof Darren Dalcher

Case study 2: NHS Direct

NHS Direct was established to provide healthcare information and advice to the public in England and Wales via a telephone helpline and online services. The creation of NHS Direct was announced in December 1997 with a target to put in place a telephone helpline by the end of 2000. National telephone coverage was achieved in November 2000. In addition, the introduction of the companion online service was scheduled for Autumn 1999 – and was met in December 1999. The NAO commented that 'given the innovative nature and scale of NHS Direct, it was a very significant achievement that both targets were met'.

Short lines of communication between the project team and those responsible for implementing the service at the local level ensured that lessons were learnt rapidly as the projects progressed. Other success factors for the project included effective use of piloting and wide-ranging consultation of key stakeholders, within the constraints of the tight schedule.

Source: NHS Direct in England, Report by the Comptroller and Auditor General, HC 505, January 2002

A familiar problem

Various studies have examined common causes of failure in large IT projects. The National Audit Office (NAO) and Office of Government Commerce (OGC), for example, have compiled a list of the eight most common causes of failure in public sector IT projects (box 3), most of which seem equally applicable to the private sector. A number of these were also highlighted in the evidence gathered in this study and are discussed in Part II.

Box 3: National Audit Office/Office of Government Commerce List of Common Causes of Project Failure

1. Lack of clear link between the project and the organisation's key strategic priorities, including agreed measures of success.
2. Lack of clear senior management and Ministerial ownership and leadership.
3. Lack of effective engagement with stakeholders.
4. Lack of skills and proven approach to project management and risk management.
5. Lack of understanding of and contact with the supply industry at senior levels in the organisation.
6. Evaluation of proposals driven by initial price rather than long term value for money (especially securing delivery of business benefits).
7. Too little attention to breaking development and implementation into manageable steps.
8. Inadequate resources and skills to deliver the total portfolio.

Source: Office of Government Commerce

Many of these causes of project difficulty seem to reflect a failure to adhere to known best practice. In the words of Martin Cobb, Treasury Board of Canada Secretariat:

"We know why projects fail, we know how to prevent their failure – so why do they still fail?" (Cobb's Paradox[9])

Possible explanations for Cobb's Paradox are discussed in Part I of the report, but best practice is already utilised in certain industries, particularly for safety critical applications. The OGC has played an important role in helping government to start to become a more effective client (see box 4), thereby initiating a market pull to raise standards in the IT supply industry. In addition, a Supplier Code of Best Practice was recently issued by Intellect[10], in association with the OGC, BCS and Institution of Electrical Engineers (IEE)[11].

Box 4: The Office of Government Commerce

The Office of Government Commerce (OGC) was established in April 2000 to improve central civil government procurement. Under the outgoing Chief Executive, Sir Peter Gershon FREng, the OGC has introduced a number of initiatives, the most important of which include:

- Identification of a named Senior Responsible Owner (SRO) to take personal responsibility for ensuring that a project meets its objectives and delivers the expected benefits;
- Gateway Reviews, which are independent reviews of projects by a team of experts at key stages, with recommendations made to the SRO.

[9] *Unfinished Voyages, a follow up to the CHAOS Report,* The Standish Group, 1996

[10] Trade Association for the Information Technology, Telecommunications and Electronics Industries in the UK

[11] *IT Supplier Code of Best Practice,* Intellect, November 2003

Furthermore, in November 2003, the Cabinet agreed six actions designed to improve the delivery of government IT projects:

- Establish Project/Programme Management centres of excellence in each department;
- Accounting Officers to provide assurance on major projects that they do not contain the common causes of failure identified by the NAO and OGC (see box 3);
- Mandate no big bang implementations and developments unless approved by a central scrutiny group;
- No government initiative dependent on new IT to be announced before analysis of risks and implementation options has been undertaken;
- Force prioritisation of all existing and new projects as mission critical, highly desirable and desirable;
- All high risk and mission critical projects to have clearly identified (i) responsible Minister and (ii) SRO and Project Manager with good relevant track records.

Source: Office of Government Commerce

Box 5: Intellect Supplier Code of Best Practice

The Intellect IT Supplier Code of Best Practice was introduced in November 2003, with input from the OGC, BCS and IEE. The aim of the Code is to facilitate a more mature IT acquisition and delivery environment. The Code includes ten commitments that cover:

- Customer-Supplier Relationships;
- Understanding the Requirements;
- Constructive Challenge;
- Confidence in Delivery;
- Assumptions and Implications;
- Programme Management;
- Risk Management;
- Supply Chain Management;
- Management and Deployment of Skilled Resources;
- Individual Skills and Professionalism.

Source: Intellect

Today's challenges

It is significant that Fred Brooks' seminal book, *The Mythical Man-Month*[12], published in 1975, contains many perceptive observations that remain disconcertingly relevant today. Nonetheless, changes have occurred that affect the nature of today's challenges. Most strikingly, improvements in computer processing power and communications technology mean that the scale of what can be attempted has escalated dramatically (see box 6). Unfortunately, developments in the design and management of complex IT systems do not seem to have kept pace with the potential of hardware, or with human ambition!

[12] *The Mythical Man-Month,* Frederick P. Brooks, Jr., Addison Wesley, 1975. Anniversary Edition, Addison Wesley, 1995

In addition, commercial pressures mean that results need to be delivered in ever decreasing time-frames, with quality often losing out in a trade-off against speed to market. Outsourcing and offshoring should enable greater access to skilled individuals and cost savings, but will also exacerbate the difficulties associated with coordination of, and communication between, team members in different locations.

Box 6: Technology Opportunities and Challenges

The exponential rate of development of computing hardware, with a quadrupling of performance every three years (Moore's Law), presents designers with obvious opportunities for building ever more ambitious systems. Applying Moore's Law, there has been a staggering **half a million fold** improvement in hardware capability in the 29 years since the publication of Brooks' book[13]. However, during that time there have been only modest increases in the power of software languages and software design and analysis tools.

The advances in computer hardware can, of course, be harvested for their potential to shrink physical size and cost rather than provide enhanced computing power. Thus we see microprocessors becoming a ubiquitous component in a wide range of engineering artefacts. A car is a good example, with dozens of microprocessors in each vehicle. Engineering success in the market place is becoming increasingly dependent on the ability to design highly reliable systems composed of sets of interacting computers operating in real time.

Whilst computer hardware progress is generally well recognised, corresponding improvements in communications technology are somewhat less widely known. In fact, the introduction of optical fibre has enabled an increase in communications speed at least as great as those in processing hardware or storage technology. The availability of such high performance at a modest cost is permitting the conception of multi-computer systems spanning intercontinental distances, with corresponding levels of design risk.

However, current software development methods and practices will not scale to manage these increasingly complex, globally distributed systems at reasonable cost or project risk. Hence there is a major software engineering challenge to deal with the inexorable rise in capability of computing and communications technologies.

In summary, there is a widespread perception that success rates for the delivery of IT projects are unacceptably low. However, good practice is routinely implemented in a number of sectors and some of the achievements in large scale IT projects are truly remarkable (see case study 2). The principles underlying these good practices should be transferable and could ameliorate the vast majority of today's problems. Nonetheless, further research into, and development of, software engineering methods will be required to meet the challenge posed by emerging classes of systems entailing complexity on a scale never encountered before.

[13] *The Mythical Man-Month,* Frederick P. Brooks, Jr., Addison Wesley, 1975

Part I: The Challenges of Software Engineering and IT Projects

An engineering discipline

"The software engineering life-cycle is very much an engineering life-cycle… I believe the principles are the same" (M. Williamson)

The systematic production and maintenance of software products, which form the basis of the IT systems discussed in this report, is commonly referred to as 'software engineering'. Indeed, the essence of both software engineering and other branches of engineering is the application of science and mathematics to find solutions to real world problems, and there are many parallels between the software engineering life-cycle and the traditional engineering life-cycle. Complex projects in both software engineering and other engineering disciplines entail specification, design, development, testing and operational support phases. They also require practices such as rigorous quality control and the sharing and management of constraints and dependencies between members of the project team.

However, as discussed above, there is a perception that IT projects have lower success rates than those in more established branches of engineering. Irrespective of the accuracy of this presumption, it is worthwhile exploring the distinctive qualities of IT projects in comparison to other engineering projects, since proper comprehension of the nature of software is prerequisite for the successful application of engineering principles to this discipline.

1. Characteristics of IT projects

Lack of constraints

"Software provides 'limitless scope' for functionality, without the inconvenient constraints of the laws of physics… There is nothing you can't do with software" (A. Bodnar)

IT projects are not subject to the laws of physics and the associated constraints in the same way as, for example, civil engineering projects. This can produce a perception that anything and everything is possible with IT. Of course, this is not the case – software is governed by real constraints, but these tend to be multidimensional and abstract in nature, and therefore difficult to understand and communicate. However, both customers and suppliers are susceptible to forgetting or simply not understanding the limitations of IT, resulting in unrealistic expectations and over-ambitious projects. Indeed, software engineers are sometimes guilty of taking on degrees of novelty and risk far in excess of the levels typically accepted by other engineers. This is due, at least in part, to the impression that software is largely free of constraints and its potential is therefore unlimited.

"Building software has been described as the ultimate exercise in 'building castles in the air'" (G. Robinson)

Case study 3: Unrealistic expectations

A prominent telecommunications company took on a contract to deliver a major complex IT system. The value they quoted to the customer was $13 million. After spending $85 million, the systems delivery team eventually admitted that they could not deliver what they had promised.

On hearing that this was the case, the customer responded by accusing the supplier of trying to get out of a difficult situation and forced them to pay $3 million in liquidated damages. In the end it became clear that the original specification was simply undeliverable and the supplier ultimately designed and delivered an alternative solution at a cost of a further $10 million.

The seeds of failure were sown at the project outset because the expectations of both the customer and the supplier were much higher than the capability of the supplier to deliver.

Visualisation

"If I was a managing director trained in law or accountancy I wouldn't ask an engineer to build a 1000 metre long concrete beam suspended at one end because I know it can't be done, I have a physical perspective about it. With software, it's never like that. We don't have any underlying feel for whether something is even feasible" (L. Hatton)

Software is effectively invisible. This visualisation problem is a source of many potential IT project failures. Senior managers commissioning IT systems may ask for functions that are over-ambitious, or even impossible to deliver, without having any sense of the level of complexity entailed in meeting their request. Moreover, unlike other branches of engineering, it is quite unusual for senior management to have significant first-hand experience of software engineering or IT project management.

It is also extremely challenging to represent the key facets of software in a way which is accessible to all stakeholders, making the specification process potentially fraught. In the case of a building, it is fairly easy to generate a physical representation that can be debated by the stakeholders and used as a blueprint. Many graphical representations are used for software specification, e.g. the Unified Modelling Language (UML), but these are subject to ambiguities and only deal with limited aspects of the system.

"It is extremely difficult to represent a specification of what you are trying to do in a precise way – even, I suspect, twins nurtured in exactly the same way would put different interpretations on the document" (J. Millar)

A further difficulty associated with the 'invisibility' of software occurs during monitoring of the project. The lack of a readily tangible product means that it is very easy for the project to proceed for a considerable time before problems become apparent, and without it being possible to verify that the passing of time and expenditure of money correlate with progression of the project in the desired direction.

Flexibility

"People believe software to be flexible, and therefore they flex it. They flex it beyond reasonable boundaries" (J. Millar)

A related problem for IT projects, also stemming from the intangible nature of software, is abuse of the perceived flexibility of software. The inability to visualise the boundaries of what is possible or practical in IT encourages people to change their mind more frequently than they might do for engineering projects where constraints are obvious. Excessive requests for new features or alteration of functions etc. during the course of

the project introduce unnecessary and undesirable complexity. This contributes to time and budget over-run, thereby increasing the chance of project failure.

Flexibility is also reflected in the fact that there are generally multiple ways of solving the same problem in software engineering, in contrast with, for example, construction where there tend to be better-defined, established processes. Moreover, whereas people are quite accustomed to having to adapt their processes to suit a building, they will often request changes to software rather than modifying their processes to fit an off the shelf, proven IT package.

Complexity

"On a large software project one is lucky if one person in 50 has anything resembling an overall understanding of the conceptual structure of the project, and divinely blessed if that person has the ability to explain it in lay terms" (G. Robinson)

Complexity can be a significant obstacle to successful design and delivery of IT projects. Although major projects in other engineering disciplines obviously also have to contend with complexity, it seems that in software engineering, complexity is both harder to detect and less well understood. In IT, complexity is multi-dimensional, encompassing scale (see case study 4), diversity, heterogeneity etc. A proportion of the complexity is warranted, i.e. necessary for the delivery of the requirements, whilst the remainder can be considered unwarranted and can interfere with the efficiency and reliability of the system.

Complexity in large scale IT systems remains an area which is insufficiently well understood. The degree of complexity entailed in achieving a particular objective can be very difficult to estimate at the project outset. As a result, projects may involve much more complexity than was originally realised; such projects are extremely susceptible to failure or difficulty. Research into methods to analyse and predict the behaviour of complex systems could lead to better assessments of the consequences of changing specific requirements during the course of the project. Technical developments in understanding complexity should therefore generate significant improvements in the design and management of complex IT projects.

"Complexity is hidden more than in a conventional engineering project" (M. Williamson)

Case study 4: Scale

Many retailers store information on what their customers purchase and use loyalty cards to identify each customer. A major supermarket chain decided to use the shopping history held on a central database to identify special offers of particular interest to each customer when they arrived at the checkout and presented their loyalty card.

In this case, the system held a record of all the items the customer had purchased during the last six months and the supermarket had approximately nine million customers every day, each purchasing an average of about 30 items. Although the project seemed attractive from a commercial viewpoint, it would have involved searching a four terabyte* centralised database, analysing the results and delivering the information remotely, almost instantaneously, to all stores concurrently. Given the existing system architecture, this was impossible! While a change to the system architecture, e.g. distributing the database to stores or caching data locally, might have made the system viable, the cost of doing so would have far outweighed the business benefit.

**A terabyte is approximately 1,000,000,000,000 bytes. If each byte were equivalent to a letter on a page, 50,000 trees would be needed to make the paper!*

Uncertainty

"The outcome of any software project is necessarily uncertain…There is no problem 'producing' software – the problem is knowing what to produce" (A. Hall)

Many complex IT systems seek to undertake or augment tasks previously carried out by people. There can be great difficulty in elucidating clear requirements for such systems. By comparison the task of actually implementing the specified system can be comparatively straightforward. There is a clear analogy here with the construction industry where there is more uncertainty in the specification of, say, a hospital than there is technical risk in building it.

"Software projects are a bit like R&D projects. It is only when you get into the project you suddenly realise it is going to take a lot longer, or cost a lot more, or there are certain features you cannot deliver, or there are certain features you cannot deliver with other features" (D. Ball)

Nevertheless, uncertainty can also cause problems in implementation of the specified system and it is possible to exceed even today's colossal computing capability. The evidence collected suggested that this is most likely to occur in meeting non-functional requirements such as security, scalability or speed of response. Limitations on the actual function undertaken by the IT system relate mainly to attempts to match human capabilities in fields like pattern recognition or natural language understanding.

Thus a significant upfront investment should be made in any complex IT system to discover and define the features of the system to be built and overcome or avoid technical limitations. This means that software projects have some of the characteristics of a high tech development project with the emphasis on early prototyping and a willingness to evolve the requirements in the light of lessons learnt.

Software and failure

"In every piece of real world software, there are embedded an unbounded number of assumptions. Most of the assumptions are not decisions that you have taken, but things that you have not thought about" (M. Lehman)

Since software is pure logic, it has no physical degradation mechanisms which can cause it to fail. In principle, this makes software very attractive – intuitively, once the designers have got it 'right', it should work correctly, indefinitely. In reality this idealistic state is never achieved and all complex IT systems have a propensity to fail. The susceptibility to failure stems from the fact that an infinite number of assumptions are embedded in every real world piece of software[14]. Most of the assumptions are not decisions that have been taken consciously, rather they arise from things that were not thought about during the development process. Unfortunately, the unbounded number of such assumptions means that sooner or later one or more will prove incorrect, potentially causing the software to fail. Even assumptions that were correct at the time of development may become invalid as the environment changes. In addition, it is very hard to predict the effects of making small changes in software and therefore to anticipate, or explain, failure.

"There are two engineering obligations with software…first of all, you design the system in such a way that failure has a minimum effect on the user, and secondly you inject enough into the design to be able to diagnose why it fails so you can correct the failure and incrementally improve the design" (L. Hatton)

This feature of software impacts greatly on the way in which IT projects must be handled. Clearly, although some element of the software may fail at some point, it should be possible to design the system in such a way that failure has no discernible impact on the user. The system should also be engineered to facilitate diagnosis of the causes of failures, in turn enabling improvements to be made to the design. Furthermore, it is perfectly possible for software to fail without the project as a whole being jeopardised if such precautions are taken. Regrettably, many projects are undertaken on the assumption that the software will be, or can be made to be, 'perfect'.

---

[14] *Software Uncertainty,* Manny Lehman, Proc. Soft-Ware, 1st Int. Conf. on Computing in an Imperfect World, Interactive Science and Tech. Conf. Centre, Belfast, Northern Ireland, 8-10 April 2002

Supporting change

"Software projects are seldom, if ever, purely for the delivery of software itself" (W. Edgar)

The majority of IT projects are undertaken to deliver some kind of business or process change. In some cases, IT systems will be introduced to enable a major business transformation, in other cases they will be automating an existing process. Even when the aim is defined as automation, the people involved will need to alter their practices, so business change in some form will ultimately result. As a consequence, IT practitioners need – but unfortunately do not always have – an understanding of the business and the processes concerned if the IT system is to achieve the intended outcome.

"The customer usually sees the problem in terms of how it has been solved or addressed in the past" (M. Lehman)

Problems also commonly arise because the description of the business process given to the supplier does not accurately represent the process being employed. In the case of automation systems, the manual process being replaced by the IT system may be intrinsically ineffective – automation is unlikely to make a bad process better, although it may execute it more quickly!

2. IT and software engineering practice

Best practice and professionalism

"In no other discipline is the gulf between best practice and typical practice so wide" (F. Brooks Jr.)

It appears that there is an exceptionally large discrepancy between best practice and common practice in IT and software engineering when compared with other engineering disciplines. There are, without doubt, a number of individuals and companies who are following best practice but unfortunately these appear to be the exception rather than the rule. Projects are often poorly defined, codes of practice are frequently ignored and there is a woeful inability to learn from past experience. In light of the increasing complexity and all-pervasiveness of software, the roles and responsibilities of software engineers are becoming more and more critical, and the need for professionalism and adherence to best practice of ever greater importance.

"The UK seems less motivated or driven to find good answers than both Americans and Asians, for example" (T. Gilb)

Interestingly, anecdotal evidence indicates that whilst other countries, including the US, encounter similar problems to the UK in the delivery of IT projects, the UK appears to be less committed to finding ways to improve performance.

"Compared with other branches of engineering in their formative years, not enough people (are known to) die from software errors. It is much easier to hide a failed software project than a collapsing bridge or an exploding chemical plant" (G. Robinson)

Software engineering is a much younger profession than most of the other branches of engineering. It has also been suggested that if IT system failures were widely known to have directly caused injury and death, there would be a stronger motivation, if not introduction of regulation, to improve levels of professional discipline. In fact, the increasing reliance on IT systems, for example in cars, means that software failure has ever more potential to result in the loss of substantial sums of money, or indeed loss of life.

Software engineering also differs from the other branches of engineering in having no tradition of professional accreditation. Chartered Engineer status is available for software engineers but there has been little enthusiasm for gaining accreditation amongst the IT community. Furthermore, unlike other engineers, software engineers enter the profession through a variety of routes, including those where no formal education in software engineering, IT or computer science has been undertaken. The

BCS is now offering the opportunity to acquire Chartered IT Professional status, which may have greater appeal to IT practitioners who do not identify themselves with traditional engineers (see box 7).

"Why, in this field apparently more than almost any other, does there seem to be no ability to learn from history?" (G. Robinson)

In addition to the factors mentioned above, a significant impediment to widespread adoption of best practice has simply been the very high market demand for IT practitioners, which, until recently, has exceeded supply. Moreover, clients are often subject to severe commercial pressures that lead them to demand delivery in the shortest possible time-frame, even if this results in a lower quality product. Together, these have diminished the incentives for suppliers to embrace professional standards and practices, although there are signs that this may be beginning to change.

Box 7: Chartered IT Professionals

Chartered Professionals are distinguished by the way they take personal responsibility for their actions and are bound by the Code of Conduct of the Professional Body which is licensed to charter them. They will normally be educated to a four year degree (or equivalent) level, having followed a course of study accredited by their Professional Body. They will then be required to complete, to peer approved satisfaction, a period of four to five years where they will need to demonstrate their technical and managerial competence. They are also required to maintain their professional competence throughout their active professional careers.

IT Professionals are required to be competent in appropriate areas including, for example, project/programme management; system design, development and operation; relationship management; security and safety; change management; software engineering; system maintenance; and quality assurance and control.

The BCS has recently pioneered the establishment of a Chartered IT Professional status to rank alongside that of Chartered Engineer, which is recognised in the longer established engineering disciplines. Privy Council recognition of the title was conferred in 2003 and the title is now open to the Professional Institutions to award.

Source: British Computer Society

"The classic engineering paradigm of control process feedback ('do not make the same mistake twice') is almost completely absent from software engineering" (L. Hatton)

Pace of change

"The pace of technological change and the ferociously competitive nature of the industry more or less inevitably lead to the triumph of speed over thoughtfulness, of the maverick shortcut over discipline, and the focus on the short term" (G. Robinson)

Another obstacle to the implementation of best practice derives from the extremely rapid pace of technological progress in IT. This makes it difficult for expertise in a particular technique or language to become established and mature, as well as creating a culture where the use of tools or solutions that are not yet proven is acceptable and commonplace (for example, products from SAP, Microsoft, Oracle, IBM and other major vendors are believed to still contain numerous bugs).

The rapidity of change in technology also tends to produce a hankering on the part of users for the most recent innovation, even if a tried and tested, mature technology could meet their needs just as effectively and more reliably.

Reinvention

"Software continues to be reinvented to perform the same function. Every time it is reinvented new errors are created" (B. Collins)

In more established engineering disciplines, the use of standard component parts, modules and procedures is routine and contributes towards the reproducibility and reliability of the systems produced. By contrast, in IT there is a strong tendency to start from scratch each time a new project is embarked on, with software continually being reinvented to perform essentially the same function. Moves towards reuse of modules and automatic generation of software have made some impact in certain industries, e.g. the communications industry, however the widespread propensity to write new code to perform well-established functions significantly contributes to a lack of reliability in many applications.

"The more one can make the switch from viewing each development as a unique project to instead being another iteration of an established process, the more repeatable and predictable the development process will be" (J. Haswell)

It is commonly argued that, unlike other engineering projects, each IT project is unique. In reality complex projects in other forms of engineering also possess certain unique characteristics. Although IT projects do tend to involve a greater degree of novelty than other engineering projects, there is no reason why many IT projects could not be approached as iterations of established processes or products with allowances made for novel attributes. In principle, repeat or routine IT projects undertaken by experienced professionals should not prove more problematic than comparable projects in other disciplines.

Management of software engineering projects

"The management processes that traditionally are applied to software projects have been derived to a large extent from those processes that were seen to be successful in other domains of engineering. It is not obvious that this historic approach to the development of software has been totally successful in spite of many attempts by talented people to accommodate its shortcomings" (B. Collins)

Despite the undeniable parallels between software engineering and other branches of engineering, there is an argument that the distinctive features of IT render some standard engineering management practices inappropriate. It has long been established that increasing team size beyond a certain number is likely actually to decrease efficiency in software engineering, due to the difficulty of sharing information between individuals pursuing creative activity in situations where their work is highly interdependent. In addition, since it is extremely difficult to define accurately the specific features of a complex IT system required at the start of the project, iterative approaches need to be invoked far more frequently than in conventional engineering projects. For example, IT projects tend to involve continuous feedback between the design and implementation stages. In general, the significance of architecture for complex IT projects also tends to be poorly appreciated.

"In a software project you don't finish any task until the whole task is complete, therefore you cannot simply decompose into a preferential sequence of events that you project manage around" (P. McHugh)

Further research into optimised management practices for software engineering projects could help to improve success rates. Nevertheless, adherence to basic, well-established engineering principles, such as careful project definition, defined decision gates and meticulous configuration management, remain key success factors that can yield enormous benefits.

Progress

"There has been phenomenal progress in the way some people write software, the way they document it, the way they handle change, the techniques, the testing, the tools" (D. Ball)

Finally, on a more positive note, there does seem to have been encouraging progress made in certain sectors of industry. The global communications backbone, for example, which contains millions of lines of software, broadly achieves requirements of two hours' total system downtime in 40 years' availability – albeit at a high cost. Similarly, some organisations in the financial sector have adopted many best practice procedures, and have exhibited corresponding improvements in project success rates. Furthermore, in aerospace, ground-based and aircraft systems achieve very low failure rates, with millions of hours between unsafe failures. Collectively, these observations provide cause for cautious optimism, demonstrating that software engineers and IT practitioners working in certain industries are already following best practice.

Part II: Building for Success

Individual companies and project teams can do much to improve their chances of delivering successful IT projects. This section provides general advice and guidance on IT projects, based on the evidence provided by contributors to the study. There do not appear to be any insurmountable barriers to adoption of this advice and many projects have already benefited from implementation of the approaches and practices outlined here. Further details of project risks and sources of best practice are described in Appendix III.

1. The Key Players

Senior management

"Without strong, sustained and high-quality leadership complex software projects are almost doomed to failure from the start and in my view should not even be commenced" (B. Collins)

Senior managers have a number of essential roles in complex IT projects, with those of senior sponsor and business customer being pre-eminent and usually undertaken by separate individuals. The *business customer* is the individual with deep knowledge of the business process being implemented, or the product being manufactured, and who is the developer and custodian of the requirements. Whilst this role is undertaken in conjunction with the other stakeholders, this is the individual whose primary responsibility is to the customer and the business.

"The project should not be seen as being driven by the IT department" (R. Butler)

The *senior sponsor* has a range of roles in a complex IT system development. First and foremost, the senior sponsor is responsible for ensuring that the project has clear high-level deliverables that relate to the overall business strategy. This is an ongoing role since the senior sponsor must continually revalidate the project in the light of evolving market or regulatory conditions. The importance of having effective leadership from a senior sponsor in the customer organisation cannot be overestimated. As discussed above, most complex IT projects involve organisational change at some level. This can be a highly disruptive experience and will only ever be achieved if the project is seen to be driven and supported by senior management – it suffers a high probability of failing if it is perceived to be sponsored by the IT department!

"Prior to commencing there should be a clear definition of the benefits to be derived from the project. Without a set of high-level objectives and deliverables a project is destined to fail before it has begun" (M. Williamson)

Furthermore, senior management needs to attempt to create a receptive context for the project. This includes, for example, putting in place performance management systems that promote constructive behaviour amongst the project team and enhancing the project management capability of the organisation. A one page summary of advice for senior managers of organisations embarking on complex IT projects is included in Part III.

End users

"Involve the 'ordinary people' in the business in setting expectations, agreeing the possible and in the daily life of the project. They know what happens now – frequently it is different from what the documented process or organisational procedures say it is" (D. Ball)

It is equally important to ensure that the end users of the IT system participate in its specification and development. There are two main benefits associated with this. Firstly, the front line users are most likely to know the exact procedures in current use and their cooperation is therefore vital to ensure that the designers receive the correct information. Secondly, it is important for users to feel a sense of ownership of the project to avoid a scenario where the end package is perceived to be imposed on them.

"The biggest problems arise when end-users specify a requirement, hand it over to a supplier and then turn up for acceptance testing, expecting the system to look a certain way – it rarely does" (P. Haren)

In addition, it is essential to devote sufficient time and attention to user training. If this does not happen, there is a danger that the IT system, and the change it represents, will be resented and resisted, potentially endangering the success of the project.

Systems architect

"It is the person who can look at a jewel and hit it just the right way so it falls into the right number of pieces. It is that ability to decompose in just the right way" (H. Lilleniit)

The pivotal technical role of the systems architect in complex IT systems is increasingly widely recognised. They produce an overview of the technical structure of the system but free from details of the implementation.

This role has been forced on projects due to the sheer breadth of hardware, software and communications options now available. Only a truly experienced and knowledgeable individual can harmonise the selections into an effective whole. A skilled architect will also produce a design which is robust, scalable and evolvable. Experienced architects should be able to incorporate sufficient flexibility to accommodate the changes in specification that generally arise during the course of projects without introducing unnecessary complexity which could compromise the integrity of the design. Ideally the scope for evolution should extend beyond the project in hand to encompass future projects or products.

"The Design Authority is the small number of architects who have the lid of the jigsaw, and it's not just about IT, it's achieving the business vision. It's these key people who understand what it is going to be like when it is all put together" (G. Hextall)

Systems architects possess the exceptional conceptual skills required to translate a business vision into a technical blueprint, which should ideally be expressed in a notation that supports formal analysis and reasoning. Moreover, systems architects must have the breadth of human and organisational understanding to address the underlying organisational and motivational issues that can critically impact on project success. The evidence we received suggests that these people can hold the key to success in a complex IT project, but are in very short supply. The UK could benefit enormously from exploring ways to identify and support people with these unique skills.

Project manager

"Regardless of the size of a project, or the number of departments or organisations involved, it is vital that there is a single project manager with ownership of overall delivery of the project on a day-to-day basis" (M. Williamson)

The project manager has overall responsibility for delivery of the project. This includes meeting the agreed budget and timescale and overseeing delivery of the specifications, testing and handover of the product to the customer. Typically, successful project managers have many years' experience – and the scars to prove it. Project management requires a broad skill set that includes leadership qualities, commercial awareness, willingness to take calculated risks, integrity, communication, persuasion and negotiating skills and problem solving ability. IT project managers additionally require sufficient understanding of the technology to identify potential difficulties arising from the distinctive qualities of IT discussed in Part I and to gain the respect of their team. The team, in turn, needs to incorporate a balanced representation of knowledge of the application area or business process, and technical and communication skills.

Whilst in engineering companies, the role of the project manager carries a high status associated with recognition of the criticality of this function to commercial success, in other sectors the project manager often seems to be undervalued. In IT there is a tendency for technical experts to be shifted into project management positions, despite the fact that project management requires quite distinct skills. Regrettably, many

"The overall project manager's role is to create the correct environment within which the team can deliver"
(M. Williamson)

organisations also lack well-defined career paths for project managers and fail to provide them with adequate opportunities for professional development. Ultimately, organisations need to realise the enormous gains to be made from developing their project management capacity and ensuring that the experience gained by their project managers is harnessed and applied to other projects within the organisation.

2. Key Success Factors

Client-Supplier relationship

"A close, open and robust relationship between buyer and supplier is key" (J. Smith)

Project success requires a constructive relationship between the customer and the supplier. This involves trust, openness and good lines of communication running between individuals in each organisation at multiple levels, from senior management through to the delivery team.

"Relationships should be as equal partnerships. Given a trustful relationship, people will come up with the archetypical: 'We could get 95% of that specification at 20% of the cost if we did such and such'"
(S. Shirley)

It is worth investing time and money in the supplier selection process. It can be difficult for customers to assess the capability of suppliers, and equally hard for suppliers who are competent to communicate their abilities to customers. ISO 9000[15] and the TickIT scheme (see box 8) provide a 'baseline' standard, while the Capability Maturity Model (CMM; see box 9) gives a progressive measure of organisational capability. TickIT and CMM are not guarantees of competence, and factors such as domain knowledge are important too. Suppliers should develop their corporate capability using schemes such as TickIT and CMM, and customers should include these schemes amongst the factors they use for selecting suppliers. Where customers do not possess the expertise required to assess supplier capability they should consider recruiting external assistance. It is certainly a cardinal mistake to select suppliers for a complex IT project on the basis of price alone. It is very difficult for suppliers to accurately predict costs at the project outset and indeed a low bid may mean that a potential supplier has misunderstood the difficulties of a project.

Box 8: TickIT

In terms of software quality assurance, the UK led the way in non-military applications through the TickIT scheme. Initially introduced with support from the Department of Trade and Industry, the TickIT software development quality system was established in 1991. The scheme is managed and maintained by the British Standards Institute with the certification processes administrated via the International Register of Certificate Auditors with the support of the BCS. All certifying bodies are required to be accredited by the UK Accreditation Service. By implementing a software development quality assurance system that gains certification, an organisation demonstrates that TickIT guidance is being followed. All UK government departments and major software purchasers recognise TickIT and it is compatible with European requirements for accredited quality system certification. Over 1,400 certificates have been issued and it is notable that half of new certificates are being granted to organisations outside of the UK.

Source: British Computer Society

[15] http://www.iso.ch/iso/en/ISOOnline.frontpage

Box 9: US Software Engineering Institute and CMM

The United States Software Engineering Institute (SEI) is a federally funded research and development centre operated by Carnegie Mellon University. It was established in 1984 through the sponsorship of the US Department of Defense, with a charter to provide government agencies with a means of assessing the capability of government contractors to provide capable and repeatable software processes. The four stated objectives of the institute are to:

- **accelerate** the introduction and widespread use of high-payoff software engineering practices and technology by identifying, evaluating, and maturing promising or underused technology and practices;
- **maintain** a long-term competency in software engineering and technology transition;
- **enable** industry and government organizations to make measured improvements in their software engineering practices by working with them directly;
- **foster** the adoption and sustained use of standards of excellence for software engineering practice.

The Carnegie Mellon SEI has been instrumental in improving IT practice in the US. One of the most important outputs has been the Capability Maturity Model (CMM), which provides a mechanism for organisations to improve their software processes. There are five maturity levels, with predictability, effectiveness and control of software processes reported to increase as the organisation progresses up the five levels. CMM Integrated (CMMI) has been introduced more recently as a process improvement framework that helps integrate multiple disciplines, with the main focus on software and systems engineering.

Source: Carnegie Mellon SEI

"I always value in business the ability to walk away from an order or project even though it is very important to you. If you have a client who is really not prepared to give the leadership, you are better off not taking the business" (D. Ball)

Suppliers are also likely to benefit from evaluating the competency of their potential customer. If the customer is not prepared to invest the leadership and effort required on their part to make the project a success, there is a strong argument for the supplier opting not to take the contract.

Moreover, if the customer is asking for something unrealistic or ultra high risk, the supplier should seek to communicate this to the customer and encourage them to review their project scope. If a customer is given such an assessment by a potential supplier, or receives few or no bids for a particular contract, they would do well to revisit their project definition. At present, it seems that neither of these is common practice, with some suppliers routinely accepting projects that they know are likely to fall foul of their allotted budgets and schedules and customers failing to take protests from suppliers about feasibility and costs of projects seriously.

Contractual arrangements

Contracting regimes can vary from fixed price, with well-defined requirements, timescale and budget, through to cost plus, where the requirements may be less clear, the timescale less well defined and the budget more flexible. The terms of the contract should reflect the uncertainty associated with a particular project and must also apportion the risks appropriately. In addition, it is vital that the contract incorporates disciplined and constructive procedures for dealing with the project if it goes off course.

"The contractual terms need to reflect the necessary uncertainty in the outcome…The contractual terms need to motivate both parties towards a successful outcome in the face of this uncertainty. In my experience, a good way to do this is through a sharing of risk and reward between the parties" (A. Hall)

Risk and reward sharing contracts can be effective in motivating both the customer and supplier to behave in a way that is conducive to success, but trust between the parties is essential. Furthermore, although the value of IT is in the business capability or advantage it gives to the customer, there is a tendency for this to be lost in detailed specifications or acceptance procedures. Where practical, i.e. the business benefit is readily measurable, the payment should therefore be contracted for in terms of delivered business benefit, rather than other deliverables.

Evolutionary project management

"Projects should always be managed by rapid learning cycles because what we are doing is so complex that nobody knows the answer to begin with" (T. Gilb)

An iterative feedback, or evolutionary, approach[16] is being increasingly frequently applied to the management of complex software and IT projects, in preference to the sequential or 'waterfall' method of project management. There is clearly still a divergence of views regarding the relative merits of the more flexible, evolutionary approach to IT project (and change) management and the more rigid, sequential model where requirements are frozen early in the project. Despite the fact that senior managers may perceive the more rigid, sequential approach to offer greater discipline and predictability, most active IT practitioners who contributed to this study favoured the evolutionary approach to project management.

Evolutionary project management uses rapid learning cycles, working on the premise that it is all but impossible to fully understand and predict the behaviour of complex IT systems at the start of the project. The overall system capability is produced in small, well-defined steps, giving users a chance to evaluate the system and influence development at a stage when the design can still be readily adjusted.

Evolutionary development naturally leads to the division of projects into phases with clear objectives, the achievement of which reduces the risk of the subsequent phases. Each project phase should feed into a project decision gate. The gates are major decision points in a project where progress and achievement are reviewed, together with plans for the next project phase. At each gate, a judgement is made about whether or not the next phase can be undertaken at an acceptable level of risk – if it cannot, further risk reduction is undertaken before the next phase is embarked upon. This approach significantly decreases the project risk by reducing the amount of work undertaken before there is a 'checkpoint', although in some cases commercial pressures may lead to multiple phases being undertaken in parallel in an attempt to accelerate the time to market.

"Successful software projects are those which are approached in a modular fashion" (B. Booth)

The first phase in the project is generally the requirements capture phase, or a proof-of-concept phase if there is uncertainty over the viability of the technology. An up-front discovery phase can also be valuable for assessing the complexity entailed in the project. For many projects, prototyping phases are vital tools for de-risking complex projects. But it should be noted that prototypes are only effective if they are treated as a means of identifying and eliminating problems. Confusing prototypes with demonstrators or pilots that are then simply scaled up will very likely impair the chances of project success. In rare cases where a 'big bang' release is absolutely unavoidable, pilots and exhaustive testing (with realistic data volumes) become indispensable.

[16] *A Spiral Model of Software Development and Enhancement*, Barry Boehm, Computer, Vol. 21, No. 5, pp. 61-72, May 1988

"The importance of feasibility exercises and concept proving to reduce risk cannot be over-estimated. Regrettably, up-front expenditure is often regarded as a waste or at best as something the supplier should be doing in his own interests. The consequential client cost associated with a supplier failure is rarely taken into account" (A. Bodnar)

Case study 5:
Integration of the Royal Bank of Scotland and NatWest Banking Systems

Following the merger of NatWest and the Royal Bank of Scotland (RBS), it was decided to migrate NatWest onto the RBS banking platform to enable a significant cost saving. This was a 16 month project but the change to all users' accounts had to be implemented over a single weekend. The programme involved the transfer of 14 million customer records, 13 million account records and 22 million direct debits, and the transformation of 250 gigabytes of data. The migration was completed on schedule and no problems of any significance occurred during or following the first day of live operation. Extensive testing with appropriately scaled volumes played a key role in the success of the project, as did strong relationships and daily meetings between senior management and the project team.

Source: BCS Technology Awards 2003

Requirements management

"Humans are very poor at saying precisely what they do want and extraordinarily talented at recognising what they don't want" (M. Lunt)

Requirements definition is one of the most critical, and most challenging, stages of the project. Many projects fail due to flaws in the elucidation of requirements, others fail because the requirements have become obsolete by the time the project is delivered.

"Do not try to achieve everything in the first implementation. Get a working system implemented – the experience of using it generally changes your view of what you want it to do" (P. Haren)

In addition to functional requirements, i.e. what the system is supposed to do, performance requirements need to be defined. Performance requirements specify qualities such as reliability, security, speed and usability and considerable effort may need to be expended in meeting these requirements for complex systems. It is also important that acceptance criteria are set during the requirements definition stage, with each acceptance procedure derived from and traceable to a project requirement. It can be quite challenging for users to know how they will recognise that a requirement has been met and qualities such as performance and security should ideally be quantified with measurable limits clearly defined in the acceptance criteria.

Change management

"Changes will occur in large complex software projects that are unpredicted. Successful projects are those that recognise that at the outset and put in place contingency processes as well as resources in funding to manage such changes in an appropriate way" (B. Collins)

"Without strict project control mechanisms, projects either never end or end up as camels which should have been horses" (D. Ball)

It is difficult to strike a balance between accepting that requirements need to be modified if the business or policy environment changes, and ensuring that excessive alterations to the scope of a project do not lead to major slippages of time or money. Requirements management is a concern throughout the life of the project and phasing and iterative approaches can facilitate this process. However, one of the great difficulties in controlling change to a project arises from the fact that it is hard to determine, and hence to communicate, the impact of a specific change on the likelihood of the project being delivered on time, to budget, and functioning as required. Judicious use of freeze dates, after which time the specification cannot be altered, can help to control project change.

It is not only essential to convey to the customer the costs associated with a particular modification of requirements, it is also vital to explain the options that will be precluded

"This change costs five tanks over the life of the project, so which would you like, this change or five tanks?" (attributed to a Ministry of Defence IT project manager)

by any particular decision. This can be a highly effective means of clarifying the customer's preferences, working on the principle that people are generally both very good at knowing what they don't want, and knowing what they do want when they are told they can't have it!

Measuring progress

"A critical success factor is a plan with milestones which are not just objectively measurable but are objectively linked to the success of the final outcome" (A. Hall)

As discussed in Part I, there are particular challenges associated with trying to monitor the progress of IT projects. A clear plan must be devised at the outset of the project including objectively measurable milestones related to the overall project deliverables. With evolutionary development, each increment must deliver real end user capability. This enables appropriate corrections to be made to plans, reducing the risk of small problems escalating into drastic slippages in terms of time and cost.

Measurement of earned value is a helpful way of ensuring that expenditure of money correlates with project progress. Earned value is the amount of money that should have been spent to produce what has been achieved – if a project is completely on target, earned value will be equivalent to the actual costs. Monitoring of earned value means that indications of problems, manifested as actual costs exceeding earned value, can be obtained at a sufficiently early stage to enable corrective action to be taken.

Case study 6: Measurement of Earned Value

A control system project was taking longer than planned to complete the requirements analysis. The supplier therefore decided to claim that the project had met its milestones without defining the human computer interface. The customer readily agreed, being keen to maintain the project schedule. As a result, the pressure was taken off the definition of the human computer interface and by the time the coding stage commenced, the interface had still not been defined. Programmers were therefore forced to make assumptions. Unfortunately, different programmers made conflicting assumptions, eventually resulting in a great deal of expensive and time-consuming work.

It was pointless to attempt to progress the design without defining the most important external interface and the reasons for the requirements analysis taking longer than expected should have been addressed directly. Measurement of earned value could have identified the problems much earlier.

Risk management

"An essential element in successful project management is identifying the risks and managing them" (J. Ferrie)

The evidence gathered in this study suggests that risk management is one of the most neglected aspects of IT project management. It requires the definition of hazards that could threaten progress, followed by brainstorming to generate a recovery plan for elimination or control of each hazard. Effective risk management is indispensable for project success: the earlier problems can be identified, the greater the chance that they can be corrected or compensated for with minimal disruption to the project. Regrettably, risk management is often limited to compilation of a risk register at the start of the project which plays little role in the day-to-day management of the project.

Management of risk involves not just identification of potential risks at the outset of the project but meticulous review of risks as the project proceeds and appropriate and timely preventative and/or remedial action. In addition, risk management continues beyond the point of delivery of the IT system – despite the costs entailed, it is often worth keeping at least part of the project team in place for several weeks or even months after the go-live date to sort out any teething problems as they arise.

Learning lessons from past projects and problems is another crucial element of risk management. This must go further than just recording 'lessons learnt' at a wash-up session at the end of each project – the lessons recorded must go on to inform the approach to successive projects. Moreover, companies should invest a great deal more effort in learning lessons from other organisations. Some companies have recognised this and developed initiatives such as 'not invented here' awards, in order to encourage staff to import good ideas from outside organisations. There are also a small number of research teams undertaking forensic studies of IT project failures, which can yield valuable information and provide opportunities for industry-wide learning[17].

"Value the experience of failure – you have just spent a fortune on a very expensive lesson" (D. Dalcher)

Case study 7: Learning lessons

A new director joined a major organisation shortly after they had recovered from a high profile failure concerning the introduction of a complex IT system. His predecessor handed him a list of lessons learnt and promptly retired. The new director was dismayed to find that the lessons listed had either been forgotten or had never been learnt by the organisation, despite the fact that the new system had only been live for three months. It appeared as if the experience of the traumatic project failure had produced no organisational learning at all.

In fact, the team who had been responsible for that project had indeed learned lessons, but the majority of them had retired or moved on, with the result that the organisation as a whole did not learn the lessons required to avoid making the same mistakes again.

Organisational culture also impacts on risk management. Effective communication between team members is essential and the culture of the project team needs to be conducive to honesty and admission of difficulty. If management appears unapproachable or intolerant, the chances are that mistakes made will not come to light until they have already manifested themselves as clear impediments to project success.

"In my experience when things go wrong there is always somebody in the organisation who knew they were going to go wrong. The question is how do you create an environment in which those who know it's going to go wrong feel able to say so and then get a proper hearing?" (G. Robinson)

[17] e.g. Groups led by Darren Dalcher, School of Computing Science, University of Middlesex and Les Hatton, Computing Laboratory, University of Kent

"We are not very good at saying that we will just take it the way it comes, because we all innately believe we are different" (A. Mather)

Novelty is another important factor that influences project risk. As mentioned in Part I, there is a great temptation to take on significant novelty in IT projects. Organisations would do well to limit the degree of novelty that they are prepared to introduce into any one project. Project risk is also reduced by employing proven and tested, reusable software components and, where possible, using off the shelf packages in place of bespoke systems. The power of commercial packages is increasing all the time and there is a persuasive argument that in most cases organisations would derive greater benefit from adapting their processes to fit a commercial package than from tailoring the package to suit their organisation.

"Packages are getting more powerful, flexible and meeting the needs of global businesses…So any business saying 'we're different' needs to take a step back and consider whether they really are different, because if global big players can manage their business with this particular package, it begs the question why can't any other business" (G. Hextall)

Software upgrades, however, were often reported to hinder projects by introducing risk. In many cases, upgrades of commercial packages offer few real benefits over their predecessors and, moreover, may contain significant numbers of bugs at the time of release, rendering them less reliable than the previous version. There is also a strong tendency to over-engineer software packages through repeated upgrades, leading to a great increase in capability that the customer does not require. Although from a risk management viewpoint there is an argument for not being in too much of a hurry to upgrade to the latest version of a new software package, end users in the organisation may feel differently!

"The use of leading edge technology always introduces significant risk into an IT project" (D. Rippon)

Testing and test planning are further crucial elements of risk management. Projects frequently appear to be meeting their budget and schedule until system testing reveals the existence of errors. Despite the extra up-front investment, the development of automated testing tools can be of great utility for complex IT systems. Unfortunately, although most plans allow time and resources for testing, they often fail to allocate sufficient time or resources to correct the errors that emerge during testing. It is therefore essential to identify and address faults early in the project, for example through effective requirements management and design reviews, and also important to realise that exhaustive testing is a practical impossibility for complex IT projects.

"Testing is an absolutely vital part of all software/IT systems projects and can often account for 40% of the overall development effort" (M. Williamson)

In summary, it needs to be recognised that thorough risk management requires investment of both time and money but can reap rewards far in excess of the initial outlay. Failure to devote resources when required is almost guaranteed to result in a much greater sum being spent at a later stage.

Technical issues

"We need much clearer stars by which to navigate and we need much better course correction instruments as we navigate towards those stars" (T. Gilb)

The evidence received clearly and unanimously identified management factors and human, rather than technical, issues as the prime causes of project failure. However, there are some key technical success factors in complex IT project delivery. A selection of some of the main technical issues is set out below.

Requirements capture seems to be seldom approached with the rigour necessary for this critical process to be accomplished effectively. The challenge entails both explicitly identifying the requirements of all stakeholders, including the customer, and expressing these in an unambiguous way that allows contradictions to be reconciled and anomalies or omissions to be identified. Wider implementation of best practice and improvements in methods, notations and tools could both help to resolve this problem.

"Few projects capture the users' requirements using notations and methods that make it possible to analyse them rigorously to find contradictions and omissions, even though these are among the most common causes of project delays and failures"
(Martyn Thomas)

Systems architecture is one of the most significant technical factors in ensuring project success. Architecture is often driven by non-functional requirements, such as performance or fault-tolerance. Design of an IT system as a set of modules with a core set forming the basic system can allow the core to be implemented and tested before additional complexity-creating modules are added. Experience shows that effective, modular systems architecture often enables projects to survive significant changes in functional requirements, and permits functionality to be delivered incrementally without major rework. At present, definition of architecture is relatively ad hoc and, whilst there is good practice, much more needs to be done to codify, validate and communicate the experience and skills of the best systems architects.

"If you have an incorrect architecture it does not matter what else you bring to the project, it will probably be doomed to failure" (H. Lilleniit)

Integration is also a serious technical problem, at several levels. At the level of computing systems and communications, multiple operating systems and protocols need to be 'stitched together' to produce a coherent overall computing platform. At the application level, software systems developed independently, including commercial off the shelf (COTS) components, need to be brought together, even though the elements may have been designed without any thought given to integration. In many cases, e.g. embedded systems, software-hardware integration can be a source of problems late in the process. Integration is an area which needs careful planning and resourcing, and a risk-driven approach to identify and tackle the most difficult aspects early on. However, developments in middleware, that is software which provides basic elements of functionality for some application domain such as customer relationship management, could significantly facilitate this process.

"I would like to solve the 'connectivity' problem… a good deal of our time seems to be spent stitching independent systems together" (M. Biggs)

Reuse of common software components has tended to be the exception rather than the rule. However, reusable modules are increasingly being employed in at least some industries and organisations – middleware and packages such as SAP are perhaps the most successful. Reuse has major advantages as it enhances the degree of reliability and predictability in complex systems. Development and more extensive utilisation of further reusable components needs to be encouraged, although there still are technical challenges to be met, e.g. in defining component interfaces, before reuse can become more widespread.

"Reuse is a fantastic thing because (a) it aids productivity, (b) it aids quality because you are reusing something that is tried, trusted, proven, and is working, and (c) it aids control"
(V. Chandler)

Verification and validation often accounts for half of software development budgets, and is usually based largely on testing. There are effective principles and practices for testing systems, including regression testing after changes. However, all too often test regimes are ineffective or are not managed in such a way that they can be repeated to show that progress is being made in removing faults from the software. Areas such as testing user interfaces, and integration testing are perhaps least well-served; in general good methods and tools exist for testing small software units. In the area of reviews, there is much well-documented good practice which is all too rarely used.

"What does make a positive difference is having a strong verification and validation programme that ensures that the code is developed and tested in line with the requirements" (A. Hall)

Part III: Findings and Recommendations

Part II outlined the measures that individual companies and project teams can take to improve the chances of project success. However, there are also actions that need to be pursued at a national level to enhance the overall environment in which IT project management takes place. Effective IT project delivery is crucial for the UK to compete internationally, for the delivery of government services, and for protecting critical infrastructure. Recommendations aimed at augmenting key elements of the national IT project management capability are listed below.

1. Professionalism

"Everybody is taught some software writing skills – they are not taught the responsibility that goes with it" (K. Longmore)

IT and software are becoming ever more pervasive and significant, with systems both decreasing in physical size and increasing in complexity (see box 6). It is therefore a matter of singular importance that those responsible for designing and maintaining these systems are deserving of the trust that is placed in them. **It is time for the IT industry to recognise collectively the engineering content of their work and to embrace the discipline and professionalism associated with traditional branches of engineering.**

Witnesses repeatedly lamented the lack of professionalism in the IT supply industry. However, persuading people to change their behaviour is not trivial. One approach would be to enforce change through regulation. IT practitioners can already attain Chartered Engineer and Chartered Scientist status from the BCS and other Professional Institutions and the new Chartered IT Professional status will be launched in April 2004. However, there is currently no 'barrier to practice' and until recently demand for IT practitioners has far outstripped supply, resulting in a lack of incentive to acquire Chartered status. While a strong track record may be the best indicator of a person's suitability to design or manage an IT system, accreditation provides a means of formalising experience and promoting a culture of professionalism.

This report does not advocate regulation of the industry as a whole at this stage. Nevertheless, there is a powerful argument that registration should be mandatory for IT practitioners working on 'high consequence' systems. These include, but are not limited to, safety critical systems; banking software, the failure of which would have enormous commercial consequences, is an example of a non-safety critical system that could be defined as high consequence.

Individual organisations, on both the customer and supplier side, also have a key role to play in promoting professionalism amongst staff and suppliers or customers, respectively. Suppliers need to be encouraged to sign up to the recently issued Intellect Code of Best Practice. Customers, on the other hand, should preferentially employ suppliers that have adopted the Intellect Code of Best Practice and insist that fully Chartered professionals are employed on important projects. Government in particular should use its purchasing power as leverage to raise standards of professionalism in the IT industry and the OGC should be supported in its efforts towards this end.

It is vital that senior management recognise their responsibility to promote professionalism in their organisation and to educate themselves accordingly. A one page summary of advice to senior managers of organisations embarking on major IT

projects is provided on p.39 as a starting point and further recommendations to address this issue are made below. However, senior managers must also seek to promote the adoption of best practice in their organisation through putting in place appropriate management practices, such as performance management systems that encourage and reward professional behaviour and providing career structures for project managers. In addition, in view of the fact that computing technology develops very rapidly, all relevant staff must be encouraged to undertake Continuing Professional Development (CPD), for example using the BCS Industry Structure Model[18].

It is therefore recommended that:

(a) Customers ensure that all senior IT practitioners involved in the design and delivery of high consequence systems have attained Chartered status and maintain their technical currency through CPD;
(b) The OGC, together with the Professional Institutions, assess means of enforcing the registration, and maintenance of professional competence through CPD, of senior practitioners working on high consequence systems;
(c) The Professional Institutions work together with the Confederation of British Industry and the Institute of Directors to promote awareness of the benefits of employing IT practitioners with Chartered status and suppliers who have adopted the Intellect Code of Best Practice.

2. Education

"We have an educational problem, not a technological problem" (L. Hatton)

A striking finding from the evidence gathered is that education, rather than technology, holds the key to improvements in software project success rates. Education is required at all levels, from senior directors to end users and delivery teams in the customer and supplier organisations, respectively. Improving education will not make an immediate change to practice, but is a vital part of a long-term solution to the problems.

University degree courses

At undergraduate level, few UK universities take an engineering approach to software engineering and computing, or address the problems of developing large-scale systems. The importance of understanding application domains, e.g. e-commerce or embedded control systems, was stressed by many of the witnesses; indeed several said that they preferred to take on staff with knowledge of the application area, rather than of computing. This is due, in part, to universities concentrating on teaching 'computing applications' such as operating systems and compilers. There is additionally a need for greater teaching of the engineering approach to software development, as well as an increased focus on 'soft' skills, such as communication.

Better use of good quality case studies could assist in familiarising students with 'real world' application domains. Additionally, some universities have established large scale complex projects, for example, extensive artificial intelligence systems that students can build onto, to provide a more realistic environment for students to gain experience of issues pertaining to complex systems. There would be benefit in more widespread adoption of this practice, for example in embedded control systems.

[18] http://www1.bcs.org.uk/homepages/514/

However, many UK universities currently lack the skills necessary to teach such courses. In addition, there is insufficient discrimination in course accreditation, which reduces the incentive for universities to alter their courses. Further impediments to change include the absence of a formal barrier to practice for IT practitioners and a lack of encouragement from employers, perhaps due to the fact that they seek people with domain knowledge rather than a computing degree. The obstacles could at least partly be surmounted through closer partnerships between companies that espouse good practice and universities which have the will, if not the expertise, to teach computing as an engineering discipline. The Professional Bodies, particularly the IEE and BCS, also have an important role to play in promoting an engineering focus in the courses they accredit.

Executive level education

Many witnesses spoke emphatically about the fundamental importance of senior management behaviour for securing project success. In today's business climate, the vast majority of senior managers can expect to encounter IT-enabled business change projects in the course of their careers. Indeed, it is highly likely that they will need to take responsibility at some point for a major investment in an IT system. Regrettably, a significant proportion of senior managers are simply unaware of the distinctive qualities of IT and their influence on project success.

Management schools could play a central role in improving senior managers' understanding of IT, both through Masters level and senior executive education. MBA programmes are highly influential in preparing future generations of managers and although many MBA programmes do include IT electives, this practice should be strengthened with IT seen as an essential component of the MBA experience. Management schools are also experienced in the provision of highly focussed courses for senior executives and their efforts in this area could be supported by the proposed UK Software Engineering Institute (see below).

"IT is now such a big part of many businesses and organisations that an understanding of it now needs to be part of the tool kit of top management" (B. Booth)

It is therefore recommended that:

(a) Intellect take a lead in forming links between companies, government departments and universities to promote a greater applications focus in undergraduate courses;
(b) The BCS and IEE produce a model syllabus or other criteria to apply in assessing and accrediting undergraduate courses to encourage a move towards courses with a stronger engineering emphasis;
(c) Management schools ensure that IT is a core module of future MBA courses.

3. Project Management

Effective project management was frequently cited as a critical factor for IT project success. It is axiomatic that this requires skilled project managers, as well as senior managers who have, at the very least, an appreciation of the importance of projects for achieving business success and a grounding in the process for running IT projects. Unfortunately, this is frequently not the case and project management is still not recognised as a mainstream career option in spite of its major role in delivering commercial success.

It is accepted that project management is difficult to teach at the undergraduate level and is best learnt through practical experience. However, management schools specialise in the transfer of experience, through methods such as case studies, and

could thus play a significant role in accelerating the acquisition of the tools and skills required for project management by both IT practitioners and the upcoming generation of senior managers. To date, project management has been poorly represented in the management schools' curricula.

It is therefore recommended that:

(a) Management schools collaborate with computer science and engineering departments, as well as the project management Professional Bodies, to develop courses which specifically address IT project management;
(b) Management schools ensure that project management is a core module of future MBA courses.

4. Risk management

A characteristic of many, although not all, failed IT projects is that they were ill-conceived or over-ambitious. In other words, their risks were not perceived at the outset and hence not properly managed. It is therefore vital that the risks of software projects are properly understood at an early stage so that appropriate risk reduction measures can be taken – which, in the extreme, may mean not embarking on the project.

The recent changes to the guidance for corporate governance, often referred to as the Turnbull Report[19], require risk management in all aspects of the business. As many large IT projects are initiated or sanctioned at Board level, the Turnbull Report gives a framework for providing guidance on IT project risk. The Royal Academy of Engineering produced a series of reports on engineering and risk in 2003, which are currently being progressed[20].

It is therefore recommended that:

The Royal Academy of Engineering produce guidance to address risks of IT projects in such a form that it can be used for corporate governance in the framework provided by the Turnbull Report.

5. Systems architects

"There are serious problems in recruiting systems architects. We will pay very good money for people who have those skills" (T. Lambertstock)

Many of our witnesses were keen to emphasise the critical role of the systems architect in complex IT projects, and the dearth of people with the relevant skills. It is likely that certain people possess a natural aptitude, including strong abstraction skills, for designing system architecture. However, it is also highly probable that their innate potential can be enhanced and developed through appropriate training. Identification of individuals with the capability to become systems architects at a sufficiently early stage in their career could be of great benefit to the UK: the role of the systems architect represents an activity towards the top of the value chain and skilled architects can command extremely high salaries.

---

[19] *Internal Control, Guidance for Directors on the Combined Code,* The Institute of Chartered Accountants, 1999

[20] Three reports on Risk: *The Societal Aspects of Risk; Common Methodologies for Risk Assessment and Management; Risk Posed by Humans in the Control Loop,* The Royal Academy of Engineering, 2003

It is therefore recommended that:

The Professional Bodies work together with the Engineering and Physical Sciences Research Council (EPSRC) and the Department for Education and Skills to explore ways to identify and develop the skills of people with the potential to become systems architects.

6. UK Software Engineering Institute

The US Software Engineering Institute (SEI) at Carnegie Mellon, renowned for its development of CMM and CMMI, has made a profound impact on standards of professionalism in the US. From the outset, the Carnegie Mellon SEI focussed on technology transfer and embedding best practice within the US IT industry, thereby raising the competitiveness of the industry and improving project success rates. The evidence collected in this study strongly suggests that there is a role for a UK institute that could provide a focal point for technology transfer and promotion of best practice, both in software engineering and IT project management in the IT supply industry, and in management in the customer community. The study also identified a mismatch between the research interests of academia and the needs of industry.

"What has interested some of the academic community most has not been what is the most urgent need of industrial practitioners" (K. Clarke)

A UK Software Engineering Institute could provide a valuable hub for academics and industrialists to join forces in identifying and promulgating best practice, as well as in addressing research problems of relevance to the UK IT industry. For example, the UK SEI could expedite the process of learning from failure at an industry-wide level by carrying out large scale forensic studies of unsuccessful IT projects. The findings could be used to direct subsequent research to the areas most needed and to facilitate industry learning through dissemination events and training.

The UK SEI would collaborate with and complement the work carried out by the Carnegie Mellon SEI and the OGC, and learn from the experience of both. In addition, the UK SEI would form partnerships with universities, companies and other government bodies, in order to enhance education and technology transfer between academia and the private and public sectors.

It is therefore recommended that:

Government, with the Department of Trade and Industry (DTI) taking the lead, work jointly with Industry to establish a UK Software Engineering Institute for research, advice and training to promote best practice in software engineering and IT project management.

7. UK Complex IT Systems Research Programme

As discussed above, the immense technological developments in processing, storage and communications technology are enabling the conception of extremely complex, globally distributed systems. Whilst some systems may already operate reliably on a global scale there has not been a systematic basis for developing, managing and evolving these systems. The Information Age Partnership[21] has already highlighted the need for research in this area and this is also one of the grand challenges identified by the UK Computing Research Committee[22]. The study agrees that this is a vitally

[21] *http://www.iapuk.org/*

[22] *http://www.nesc.ac.uk/esi/events/Grand_Challenges/*

"The challenge for software engineers in attempting to provide solutions for large complex problems is that the complexity of the solution itself is poorly understood" (B. Collins)

important area for the UK – both in terms of the UK IT industry and the services on which much of the UK economy depends. Establishment of a UK SEI would provide a natural base for a research programme in this area.

A key objective of this programme would be to give the UK an understanding of how to develop and assure the critical IT infrastructure which is progressively being deployed. Examples of specific IT complexity challenges that need to be addressed include:

- Software components integration
- Service integration and design for reduced risk
- Trust, integrity and reliability
- Knowledge management
- Autonomic computing
- Collaborative organisational models
- Socio-political implications of large-scale and/or pervasive computing systems

It is therefore recommended that:

The DTI and EPSRC establish a UK research programme on complex IT systems to address the design, development, evolution and assessment of complex, distributed IT systems.

"With tens of billions of pounds wasted annually on poor quality software, and failing projects in the news most weeks, investment in computer science and software engineering could have an enormous payoff for the UK economy." (Martyn Thomas)

Advice for Senior Management: Five Key Issues

1. The Project

Before embarking on any major IT project, ask yourself the following questions:

- *Why are we doing this?*
- *What contribution will this make, not just to a specific problem but to the overall business strategy?*
- *Am I, as the CEO/IT Director/Finance Director, prepared to devote enough time in my 80 hour week to the understanding of what this project is and to monitoring its progress?*
- *Are my expectations of this project, including time-scale and budget, realistic?*
- *Have we clearly and unambiguously recorded the ten most important requirements for this system?*
- *How important is it that we do this, given that it will cost a huge amount of money, time and disruption?*
- *Can we sustain this with all the other things that are going on as well?*
- *Am I confident that I have people with the right skills, and knowledge of IT and the application, who I can trust to handle this project?*
- *Have we done something like this before?*
- *Do we understand the risks involved with this project?*
- *What are we doing to manage those risks?*
- *How would this project be affected if the market changed, or a new technology became available?*
- *Do I have a recovery mechanism if everything goes wrong?*
- *Am I managing this organisation in a way that is conducive to project success?*

If you do not feel qualified to answer these questions, seek expert help. If necessary, employ an independent review team to assist in addressing these questions and monitoring progress.

2. The People

Ensure that three individuals, not committees, are publicly identified by name:

- The overall executive sponsor who will receive the glory for success or memorial for failure of the project;
- The *systems architect;*
- The *project manager.*

3. The Benefit

Make sure that your staff can clearly define the business benefit that will be delivered by the project – this must be absolutely comprehensible and defensible. If different key staff members give you different answers to this question, you have cause for concern. Also ensure that you know how delivery of the business benefit will ultimately be assessed, and who will take responsibility for that process.

4. The Complexity

It is generally less important that you understand how the benefit will be delivered, i.e. the details of the software. However, a general rule is 'keep it simple'. This means avoiding big bang project implementations unless essential and being willing to use off the shelf packages where possible, adapting your business and management processes to fit the system rather than tailoring the system to suit your organisation. Although this may seem an unattractive compromise, the chances of achieving a successful outcome will be drastically improved.

5. The Progress

Finally, throughout the course of the project, ask your staff what worries them most about the project and what could go wrong. Their answers are likely to be much more informative than if you ask them whether everything is on track!

Appendix I: Membership of the Working Group

Chairman:

Basil R. R. Butler CBE FREng FIMMM
Formerly Managing Director, BP Plc

Members:

Peter K. Blair OBE FREng FIEE
Formerly Technical Director, Racal Defence Systems Ltd

Dr Allan J. Fox FREng FBCS FIEE
Formerly Managing Director, Harlow Laboratories, Nortel Networks

Dr Ian A. M. Hall FREng FRAeS
Formerly Engineering Director, British Aerospace

Keith N. Henry FREng FICE
Formerly President and Chief Executive, Kvaerner E&C Plc

Professor John A. McDermid FREng FBCS FIEE FRAeS
Professor of Software Engineering, University of York
Non-Executive Director, High Integrity Solutions Ltd

Dr John Parnaby CBE FREng
Former Group Director, Lucas Industries plc
Chairman, Aston Academy of Life Sciences Ltd

Secretariat:

Dr Hayaatun Sillem
Assistant Manager, Engineering Policy,
The Royal Academy of Engineering

Dr Mike Rodd FBCS FIEE
Director, External Relations, Knowledge Services and Forums,
British Computer Society

Appendix II: List of Contributors

Tom Abram	Mantix
Dr John Arthur FREng FIEE	Thales-MESL Ltd
David Ball FREng FIEE	ECI Telecom Ltd
John Baxter FREng FIEE	Powergen UK Ltd
Prof Alistair Bellingham CBE	NHS Information Authority
Prof Ralph Benjamin CB FREng FIEE	Formerly: GCHQ
Matthew Biggs	Siemens Financial Services Ltd
Prof Phil Bennett FREng FBCS FIEE	CSE International Ltd
Adam Bodnar FREng FIEE	Formerly: INBIS Group Plc
Dr Ben Booth FBCS	MORI
Prof Peter Brook FREng FIEE	Defence Procurement Agency
Martin Burstyn	Consultant
Prof Fred Brooks Jr. FREng DistFBC	University of North Carolina at Chapel Hill
Richard Butler	Renishaw Plc
Vince Chandler	CitiGroup
Keith Clarke FREng FBCS FIEE	Formerly: BT
Nick Coleridge	Consultant
Prof Brian Collins FBCS FIEE	Royal Military College Shrivenham
George Cox	Institute of Directors
Prof Jon Crowcroft FREng FBCS FIEE	University of Cambridge
Prof Darren Dalcher	University of Middlesex
Bill Edgar CBE FREng	Wood Group
William Everitt FREng FIEE	Domino Printing Sciences Plc
Dr John Ferrie FREng	Smiths Group Plc
Prof Anthony Finkelstein FBCS FIEE	University College London
Dr Pat Foster FREng FIEE	Microwave & Antenna Systems
Kay Gamble	Barnardos
Dr Colin Gaskell CBE FREng FIEE	Ferranti Technologies Ltd
Sir Peter Gershon CBE FREng FIEE	Office of Government Commerce
Tom Gilb	Consultant
Dr Jonathan Grant	RAND Europe
Dr J Anthony Hall FREng FBCS	Praxis Critical Systems Ltd
Dr Patrick Haren FREng FIEE	Viridian Group Plc
Jon Haswell	IBM Research
Prof Les Hatton FBCS	University of Kent
Gordon Hextall CB	NHS IT Programme
John Holt FREng	Surrey Satellite Technology Ltd
Nigel Hughes FREng	Airworthiness Requirements Board
John Ivinson FBCS	Consultant
Chris Jack FIEE	Rolls-Royce
Robert Kennedy	National Audit Office
Hermann-Josef Lamberti	Deutsche Bank
Tim Lambertstock	BACS
Manny Lehman FREng FBCS FIEE	University of Middlesex
John Leighfield FBCS	Research Machines
Harry Lilleniit	Nortel
Prof Bev Littlewood	Centre for Software Reliability
Keith Longmore	Lotus
Morcom Lunt FIEE	CSC Computer Sciences Ltd
Alan Mather	Office of the e-Envoy
Patrick McHugh	The Trinity Group

Sir Duncan Michael FREng	Arup Group
John Millar FBCS	Charteris Plc
Rod Muttram FREng FIEE	Bombardier
Dr Ian Nussey OBE FREng FBCS FIEE	IBM University Relations
Prof John O'Reilly FREng FBCS FIEE	Engineering and Physical Sciences Research Council
Dr Sarah Pearce	Parliamentary Office of Science and Technology (now at GridPP, Queen Mary, University of London)
John Ponting	Met Office
David Rippon FBCS	BCS Elite Group
Dr Geoff Robinson CBE FREng FBCS FIEE	British Geological Survey
Dr Chris Sauer	Templeton College, Oxford
Dame Stephanie Shirley OBE FREng FBCS	Xansa Plc
Mr Ian Shopland	BT Exact
John M Smith FBCS FIEE	IBM
David Taffs	Arup
Mark Thomas	IBM
Dr Martyn Thomas FBCS FIEE	MTA Ltd, IEE IT Sector Panel
Alan Vincent MBE FREng	SBAC Technical Board
Sir Robert Walmsley KCB FREng FIEE	Formerly: Chief of Defence Procurement, Ministry of Defence
Nick Wensley	World Class International
Dr Mark Williamson	Accenture
Dr Rob Witty FBCS FIEE	NATS
Nick Woodward	PA Consulting

Appendix III: Project Risks and Sources of Best Practice

IT systems are very diverse so it is not straightforward to identify areas of risk and best practice which apply across a wide range of projects. The aim here is to highlight key issues which are likely to need attention in most projects. It is intended that the UK SEI, if established, would develop and maintain much more comprehensive guidance on best practice (see recommendation 6); The Royal Academy of Engineering is addressing risk issues (see recommendation 4).

General

Fred Brooks' *Mythical Man-Month*[23] remains the most readable account of the problems of large scale software development. Despite its age it contains observations and advice of great relevance to current projects, which in itself is telling.

Managerial Considerations

There are a number of existing sources of best practice and other guidance:

- Simple guidance for top-level company management is provided on p.39. *Senior company managers or directors should ask themselves these questions at the start of each major IT project.* More detailed guidance is provided in *Managing by Projects for Business Success* by John Parnaby, Stephen Wearne and Ashok Kochhar[24].
- *Troubled IT Projects* by John M. Smith gives extensive guidance on how to identify and control project risks[25]. *Managers*[26] *should consult this book periodically, especially at major reviews and times of project crisis.*
- The OGC Gateway process gives guidance on management of public sector projects, particularly those which are IT intensive[27]. Whilst these guidelines were not developed for commercial projects, much of the guidance would transfer directly to this domain. *Managers, whether in the public or private sector, should consider applying the OGC framework to their projects – and be prepared to justify their actions to their senior management if they have not employed this process.*
- The Institute of Chartered Accountants give comprehensive guidance on legal risks associated with IT projects[28]. *Managers should consult their legal experts, as well as this useful overview of the issues, when constructing a project risk register.*
- There is overwhelming evidence that incremental developments are much less risky than 'big bang' projects. Various authors, e.g. Gilb[29], give advice on incremental software development projects. *Managers should use incremental processes wherever possible – and be prepared to justify their actions to their senior management if they use a "big bang" approach.*

Technical Considerations

A number of best practice guides and standards exist but these have become outdated as the discipline is moving so quickly. The STARTS Guides[30] were very valuable in their day and

---

[23] *The Mythical Man-Month*, Frederick P. Brooks, Jr., Addison Wesley, 1975. Anniversary Edition, Addison Wesley, 1995

[24] *Managing by Projects for Business Success*, John Parnaby, Stephen Wearne and Ashok Kochhar, Professional Engineering Publishing, 2003

[25] *Troubled IT Projects,* John M. Smith, IEE Professional Applications of Computing Series 3, 2001

[26] This refers to managers of major IT projects; the term managers is used throughout this appendix for brevity.

[27] Information can be found at www.ogc.gov.uk

[28] *IT Legal Risk Management*, Rachel Burnett, Institute of Chartered Accountants, 2003

[29] *Principles of Software Engineering Management*, Tom Gilb, Addison-Wesley Longman, 1989

[30] e.g. *STARTS Purchasers' Handbook "Procuring Software-based Systems"*, Second Edition, prepared by industry with the support of the Department of Trade and Industry and the National Computing Centre, 1989

some of the information, e.g. on requirements and configuration management, is still pertinent. There are some comprehensive reference books[31] and undergraduate texts[32], but in the absence of a suitably concise, up-to-date source of technical best practice, a series of questions are posed below which highlight different areas of technical risk. If the answer to any of the questions is 'no' then this indicates an area of project risk which needs to be mitigated.

- Requirements – *are all stakeholders involved in determining and validating requirements? Are requirements animated or exercised in support of validation? Have they been analysed rigorously to identify omissions and to remove contradictions? Are non-functional properties, e.g. usability, performance, defined? Are the requirements objectively testable? Are requirements prioritised in terms of their value to the customer? Have likely changes been identified and recorded?*
- Architecture – *are there means of modelling and analysing key properties of the architecture, e.g. throughput, resilience to faults, resource usage? Is it possible to map, or decompose, requirements to individual elements in the software architecture? Can the components be implemented at reasonable risk? If the components are implemented as specified will the system meet the requirements? Is the architecture animated or exercised in support of validation? Can the architecture accommodate the identified changes?*
- Detailed design and implementation – *are notations and programming systems used which facilitate verification that the software meets its requirements?*
- Verification – *are all specifications, items of software, test scripts, etc. reviewed by people other than their authors? Have the designs and items of software been analysed by tools that can detect errors and inconsistencies? Are individual software components tested by people other than their authors? Is there a systematic way of ensuring that all the software in individual components is thoroughly tested? Is there a way of systematically testing the software as it is progressively integrated to produce the complete system? Is there a means of regression testing following changes (to ensure that changes have not caused unintended changes in system behaviour)?*
- Reuse – *is software reused where possible, e.g. by migrating code from legacy systems, by using packages such as SAP[33], WebSphere[34], or by using automatic code generation?*
- Traceability – *is all software traceable to a top-level requirement, and are all requirements traceable to the software which implements them?*
- Configuration management – *is it possible to identify every software item in a particular build, or release, of a software system? Is it possible to re-build any previous release of a software system? Is it possible to know to which item of software a user problem report relates?*
- Change management – *is it possible to determine the impact on the software and all associated information (requirements, architecture, tests) of any proposed change before implementing it? Are changes grouped to minimise their impact and to ease their verification?*
- Tools – *are tools employed judiciously to remove drudgery from software engineers and to reduce the chance of errors, whilst giving them the freedom to create effective designs? Are tools integrated to maximise the efficiency with which the software process is conducted, and to minimise the risk of errors of transcription in moving data between tools?*
- Standards – *are all relevant standards used to ensure reliable specification, integration and application of best practice, organisational structures, methodologies, tools and techniques?*

---

[31] e.g. *Software Engineers Reference Book,* J. A. McDermid (ed), Butterworth Heinemann, 1991

[32] e.g. *Software Engineering,* Ian Somerville, Pearson Education, 2000

[33] http://www.sap.com/

[34] http://www-306.ibm.com/software/info1/websphere/index.jsp

Appendix IV: Glossary and Abbreviations

Application domain	An area of application of computer systems, e.g. embedded control, e-commerce, banking. Often used to identify the body of knowledge necessary to understand the requirements for the system.
Application software	Software which provides 'end user' functionality such as word processing, air traffic management or process control.
Autonomic computing	The creation of computer systems that have the ability to self-manage and self-heal. The term 'autonomic' comes from the autonomic nervous system, which controls many organs and muscles in the human body.
BCS	British Computer Society
CEO	Chief Executive Officer
CMM	Capability Maturity Model; a model for judging the maturity of the software processes of an organisation and assisting in process improvement.
CMMI	Capability Maturity Model Integrated; a process improvement framework that helps integrate multiple disciplines, especially systems and software engineering.
COTS	Commercial Off The Shelf
CSF	Critical Success Factors
Decision gates	Major decision points in a project where progress and achievement are reviewed, together with plans for the next project phase. If it is decided that the next phase cannot be undertaken at acceptable levels of risk, further risk reduction must be carried out before the project is allowed to progress to the next phase.
DTI	Department of Trade and Industry
Earned value	A method of measuring project performance that uses original estimates and progress to date to determine whether the costs incurred are on budget and tasks are being accomplished on schedule.
Evolutionary project management	An iterative feedback approach involving incremental delivery of functionality, and rapid learning cycles which result in clearer understanding of the objectives for the next project iteration.
EPSRC	Engineering and Physical Sciences Research Council
FBCS	Fellow of the British Computer Society
FICE	Fellow of the Institution of Civil Engineers
FIEE	Fellow of the Institution of Electrical Engineers
FIMMM	Fellow of the Institute of Materials, Minerals and Mining

FRAeS	Fellow of the Royal Aeronautical Society
FREng	Fellow of The Royal Academy of Engineering
Hardware	The physical components of a computer system, including peripheral equipment such as printers.
High consequence systems	IT systems where failure could have serious health, safety or commercial consequences.
IEE	Institution of Electrical Engineers
Industry Structure Model	British Computer Society database of over 300 IT roles, nine function groupings and ten levels of responsibility used to help identify staffing requirements, create job descriptions, assess staff competences and establish training requirements.
Intellect	Trade Association for the Information Technology, Telecommunications and Electronics Industries in the UK
ISO 9000	An internationally accepted set of standards that provide a framework for implementation of a quality management system.
IT	Information Technology
MBA	Master of Business Administration
Microprocessor	An integrated circuit that contains the entire central processing unit of a computer on a single chip.
Middleware	Software package that simplifies construction of applications, e.g. for business process automation.
Moore's Law	The observation made in 1965 by Gordon Moore, co-founder of Intel, that the number of transistors per square inch on integrated circuits had doubled every year since the integrated circuit was invented. Moore predicted that this trend would continue for the foreseeable future. In subsequent years, the pace slowed down a bit, but computer processing power has doubled approximately every 18 months, and this is the current definition of Moore's Law. Most experts expect Moore's Law to hold for at least another two decades.
NAO	National Audit Office
OGC	Office of Government Commerce
Operating System	Software, such as Windows, Unix or MacOS, which manages the underlying machine and provides basic facilities such as file storage for users and applications software.
R&D	Research and Development
RBS	Royal Bank of Scotland
Requirements	Define what the system is intended to do and the constraints under which it is required to operate.

SAP	A suite of business process automation products and services produced by SAP GmbH.
SEI	Software Engineering Institute. Usually refers to the research institute at Carnegie Mellon University responsible for many innovations in software engineering, most notably the CMM.
Software	A computer program which provides the instructions that direct the operation of the computer hardware. Software is often divided into operating systems (qv) and applications software (qv). Recently the term middleware (qv) has been used to refer to any layer between the operating system and application.
SRO	Senior Responsible Owner
Systems architecture	The definition of a unified and coherent structure for the system, consisting of its constituent parts and the connections that establish how those parts fit and work together. Effective systems architecture is essential for complex projects.
TickIT	TickIT is a quality assurance certification scheme developed to apply ISO9000, but adapted to deal with the special requirements of software development. Certificates are administered through the International Register of Certificate Auditors, with the support of the British Computer Society.
UML	Unified Modelling Language. A widely used set of graphical notations for specifying software, see for example *Unified Modelling Language Reference Manual*, James Rumbaugh, Ivar Jacobson, Grady Booch, Addison Wesley, 1999.