December 2001

Why software engineering is riskier than ever

A. Fuller
University of Wollongong, annef@uow.edu.au

P. Croll
University of Wollongong

O. Garcia
University of Wollongong

Why software engineering is riskier than ever

Abstract

The ultimate aim of software engineering methods and tools is to reduce risks associated with the development of software products. Accepted risk management procedures recognize that software development is inherently risky, however fail to take into account the changing nature of both software products and the organizations undertaking their development. SE research has traditionally focused on the needs of very large corporations undertaking equally mammoth and complex development projects, thus most tools and methods are predicated on this model. Yet most software development is undertaken by small to medium enterprises. Consequently, these development efforts are either undertaken with little or no adherence to any SE standard process or by attempting to tailor processes intended for larger organizations. Neither of these alternatives is ideal, with both introducing new elements of risk. The rise of the Internet as a platform for commercial applications has partly driven this move away from monolithic software development and has also contributed to the proliferation of software products built using COTS components. Clearly there is a risk involved in the use of components not developed specifically for a particular application while Internet applications introduce an. entirely new set of possible risks. These newer risk factors directly affect the quality of the software product and the paper discusses these factors in detail, showing how they contribute to making software development today an even riskier proposition than previously.

Keywords

Internet, bibliographies, project management, risk management, software development management, software quality

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Fuller, A, Croll P & Garcia O, Why Software Engineering is Risker Than Ever, Proceedings of the Second Asia-Pacific conference on Quality Software, 10-11 December 2001, 113-119. Copyright IEEE 2001.

Why Software Engineering is Riskier than Ever

Anne Fuller, Peter Croll, Omar Garcia
University of Wollongong
{annef, croll, ogarcia}@uow.edu.au

Abstract

The ultimate aim of Software Engineering methods and tools is to reduce risks associated with the development of software products. Accepted risk management procedures recognize that software development is inherently risky, however fail to take into account the changing nature of both software products and the organizations undertaking their development.

SE research has traditionally focused on the needs of very large corporations undertaking equally mammoth and complex development projects, thus most tools and methods are predicated on this model. Yet by far the majority of software development is undertaken by Small to Medium Enterprises. Consequently these development efforts are either undertaken with little or no adherence to any SE standard process or by attempting to tailor processes intended for larger organizations. Neither of these alternatives is ideal, with both introducing new elements of risk.

The rise of the Internet as a platform for commercial applications has partly driven this move away from monolithic software development and has also contributed to the proliferation of software products built using COTS components. Clearly there is a risk involved in the use of components not developed specifically for a particular application while Internet applications introduce an entirely new set of possible risks. Indeed, depending on the nature of the application itself many of these products can be described as "critical" and are therefore candidates for formal risk assessment procedures.

These newer risk factors directly affect the quality of the software product and this paper discusses these factors in detail, showing how they contribute to making software development today an even riskier proposition than previously.

1 Introduction

The discipline of Software Engineering (SE) was born in 1968 into a world in which the software development was generally concerned with the production of custom software under contract for large corporations such as the United States Department of Defense (DoD). Both governments and industry recognized that a major factor underlying the problems associated with developing software was an inability to manage the process, and software engineering was intended to overcome this limitation by providing methods and processes for the timely construction of large, quality software systems [23]. Subsequently, software development methodologies were created to offer an "engineering-like" development environment, with the adoption of a software development methodology seen as a major factor in reducing the risks associated with shortcuts and mistakes and ensure the quality of the software product (Roberts, 1999).

Much SE research has been sponsored by the larger corporations in a bid to address problems associated with the contract model of software development [16]. For example the Software Engineering Institute (SEI) has the support of the DoD while the European Software Institute (ESI) has the backing of big names like SAP AG, Honeywell, and CERN. Such corporations have the luxury of access to virtually unlimited resources. Although SE research has produced processes and tools aimed at producing software more likely to perform as required in less time and at lower cost, this has mainly benefited those few companies large enough to take advantage of these advances [36].

In this paper we show that most of today's software is being developed by Small to Medium Enterprises (SMEs) rather than large companies. This results in the situation where most software development projects not only face the risks normally associated with any business project, but also additional risks introduced by the necessity to adapt processes developed without consideration for the constraints and difficulties confronting smaller enterprises.

We will also show that the changing nature of application domains, particular the rise of web-based applications, means more and more software faces hazards formerly confined to safety critical software, while the trend toward constructing software from components adds yet a further dimension to the risks inherent in today's software projects.

We conclude that the majority of today's software projects are riskier than ever.

0-7695-1287-9/01 $17.00 © 2001 IEEE

2 SME's and Software Development

The Australian Bureau of Statistics, quoted in [19] defines a "small business' as one employing fewer than 20 people while a "medium business" is a business that is not a small business having net assets less than $200 million and employing fewer than 200 Full Time Equivalent persons. Combining these two descriptions results in the definition of a Small to Medium Enterprise (SME) as a business having net assets less than $200 million and with less than 200 employees and less than $200 million in assets.

The microcomputer revolution has changed the software landscape with much of today's software being produced for the mass market. With fewer than 6% of all software houses in the United States having more than 50 employees, [16] claim that the majority of software shops, the SMEs, are in the situation of having to adapt processes which do not address important development issues of company size, development mode, development size and development speed.

There is evidence that these figures are similar in other countries. For example, [21] state "in Germany many software companies are small or medium-sized, as are most software development departments in larger companies." Although we have no direct figures for Australia, the Australian Bureau of Statistics (quoted in [1]) reports that 96% of all businesses in the information and communication technologies sector employ fewer than 20 persons. In addition, a report prepared by SMT Consulting for the Australian Department of Industry, Science and Tourism states "There are over 9000 computer specialist firms employing 55,000 people in Australia. A large proportion of these would be involved in software development" [50]. Therefore, on average, software development houses in Australia comprise just over 6 employees. Another census [10], reports that of 64 software related companies interviewed in a third world country, only 3 employ over 50 developers.

3 SMEs and Risk

An SME's exposure to risk is multi-faceted. In addition to project management or business risks, the SMEs software developer must also deal with risk associated with their development process and with the need for greater attention to integrity levels brought about by changing nature of the product.

While any software development project involves uncertainty [46], the SME is even more sensitive to these risks. For example the small startup company producing shrink-wrap software must first develop the product, before finding actual, paying customers [16]. If the product fails to attract a market, the company itself is likely to fold. A risk analysis may be carried out for these types of business risk, either formally or informally, with the assessment always in monetary terms. In addition, one of the key driving forces for new products is time-to-market. Small enterprises need to weigh the risks of any delays in launching the product against releasing a product before it is sufficiently mature.

There is a trend toward smaller companies developing software in environments where recommended SE practices are not always applicable, and therefore practiced selectively. Very few SMEs have the resources or time needed to adopt any of the theoretical SE processes [16], [36]. Instead, they often rely on intuitive reasoning, using processes that are sometimes based broadly on SE principles, but more commonly are quite unstructured [16], [36].

Simply "scaling down" SE processes is unlikely to succeed. [5] claims that scale only affects management complexity, that software development effort has always been modularized and therefore SE practices are focused on the work of the individual and thus on "SE in the small". This ignores the difference in structure of large and small organizations. For example large organizations can afford multiple specialists, whereas smaller groups must rely on generalists [24]. This lack of specialization may in itself pose a risk if the success of a particular project is reliant on some core competency. (Alternatively, as Laitenan [24] points out, reliance on a specialist can itself be risky. What happens if the specialist leaves?)

In the small group one person will fulfil a multiplicity of roles that in a larger group would be assigned to different individuals. This affects the way in which the system is decomposed, thus the modularization of tasks for a smaller group will necessarily differ from that in the larger organization and therefore operations to be carried out will also differ [25]. As an example of this difference, Laitenan [25] cites the traditional code inspection involving at least seven people in a face-to-face meeting is not possible in a group of four, with some members telecommuting.

Boddie [6] claims this is just an example of management decreeing the use of an inappropriate methodology, and is a management failure, not an SE failure. In such situations SE processes are never "scaled down"; if inappropriate the method is ignored.

Fayad [16] call for "tailored software engineering" practices for these smaller organizations. A number of other authors have discussed the need for SE processes to be made more relevant to the majority of today's software developers i.e. SMEs, and have enumerated requirements for "SE in the Small" , [16]. Indeed recently published approaches such as Beck's Extreme

Programming [4] have been developed to better suit the SME.

However we suggest a "one size fits all" approach for SE in the small is as fraught with the possibility of failure as attempting to scale down from SE in the large. Instead, this now significant proportion of the software industry needs mechanisms allowing them to do their own tailoring.

Pressman's Adaptable Process Model (APM) is promising in that it "is intended as a basis from which a customized software process can be developed" [42]. It provides a checklist of adaptation criteria to enable teams to adapt the APM to meet their specific project needs, depending on the *degree of rigour* with which the software process will be applied. Although the APM addresses Risk Management as an element of the process, there is little guidance as to deciding what degree of rigour to apply, nor is there a process for determining the risks associated with reducing rigour in any given situation.

Clearly, the decision to reduce or omit entirely any SE practice increases risk exposure. SME software manages need to know how to assess the risks associated with decisions regarding method, in order to select wisely. This cannot be looked up in a book or chart as "we do not know ... which software engineering technologies work best in which situations" [39].

4 Integrity Risks

In addition to the usual business risks and risks related to choice of method, software development projects involve a third element of risk. The growing dependence on computers for business and life-critical functions requires that many applications offer some guarantee of integrity and that risk techniques formerly confined to safety critical systems are now more broadly applicable [33]. For example, the growing reliance on Internet based software leads to increased security hazards, E-commerce solutions require demonstrated levels of security, while the medical profession's growing reliance on electronic medical databases not only requires security guarantees but introduces safety hazards as well.

4.1 Web-based Software

Businesses, governments, other institutions and individuals now use the Internet for purchasing, sales, and communications and personal development or recreation. All expect that the systems with which they interact be both reliable and secure. [48]

In November 1988, the so called "Internet Worm" halted 10% of Internet systems. Following that incident, the US Defense Advanced Research Projects Agency (DARPA) charged the Software Engineering Institute with setting up a Computer Emergency Response Team Coordination Center (CERT/CC), to coordinate communication among experts during security emergencies and to help prevent future incidents [11][57]. Since then, the CERT/CC has helped to establish other response teams and their incident handling practices have been adapted by more than 90 response teams around the world [11].

The CERT/CC 2000 Annual Report states that the organization received 774 vulnerability reports and handled 21,756 computer security incidents affecting more than 9,350,0001 sites last year [12]. These statistics indicate that security remains a fundamental issue for developers of web-based applications.

According to [35], the use of networked systems to improve organizational integration increases the risk of system intrusion and compromise, however such risks can be mitigated by adding survivability capabilities i.e. the ability to preserve essential services in the face of a security breach. They propose the integration of survivability into the software development cycle as a means ensuring systems can withstand failures.

Reliability is a separate concern. As more and more everyday tasks are performed using the Internet, both individuals and businesses are becoming increasingly dependant on the tools and services the Internet provides [51]. Many start-up Internet companies are founded on promoting the use of such tools, thus are basing their future on the reliability and usability of the applications as perceived by users. Applications with lengthy load times, too many features or just plain errors adversely impact customer loyalty [40]. There is a concern that, in the rush to get a product online, quality may be compromised, thus increasing the risk that a service is either insecure or unusable [40],[22].

Newsom *et al* [38] have even proposed that the Internet can be a "valuable component of emergency preparedness", and has listed a number of incidents where the Internet was used as a means of communicating details of the emergency to the wider community. Such usage exploited existing technology to provide information. However, the authors further propose that the Internet could support command, control and surveillance tasks in emergencies, with clear implications for integrity requirements of the applications involved.

4.2 E-commerce

The discussion of E-commerce is intentionally separated from web-based software in order to highlight some particular needs in isolation from general web-based applications.

E-commerce is defined in a number of different ways from the simple "trading via the Internet" [12], to "the integration of internal and business processes through information and communication technologies" [2]. Whether we restrict our concern to internet-only transactions, or take the broader view, the adoption of an e-commerce application introduces new risks. In addition to the reliability and security issues already discussed, the e-trader must establish a relationship with customers based on trust [2], [27].

Yan & Paradi [55] suggest that in order to encourage customers (to Internet banking) they must be confident the communications and transactions are secure, that the origin of messages can be verified and are legally binding and that personal privacy can be protected. Ahuja [2] expands this list of ingredients for building mutual trust to five:

- strong authentication of the identity of both parties,
- nonrepudiation of transactions to counter the threat that a party may deny the transaction took place,
- confidentiality of transactions to protect against unauthorised disclosure of transaction details,
- privacy to protect information about individuals and
- data integrity to ensure no distortion, modification or loss of data.

However there is still a lack of e-commerce standards accepted by either businesses and consumers [56]. The risk for the business is two-fold. As there is no standard for ensuring security of transactions, what assurances do they have that the system is secure and how can they inspire potential customers to trust the system. Organizations adopting an e-commerce application and their customers need to be reasonably assured that the application offers adequate levels of security for such transactions. Application developers should be able to assure the integrity of their e-commerce solution, particularly in regard to the essential elements for trust.

4.3 Health

Medical databases are proliferating, creating a potentially hazardous environment [18]. With the safety of patients at risk it becomes even more critical the software developer be familiar with risk assessment techniques. For example, with a distributed database of medical records a risk analysis may identify that incorrect linkage to somebody else's medical record may be considered catastrophic. The incorrect link is itself not necessarily harmful provided data in the original record does not get corrupted. The hazard occurs when a patient gets incorrectly diagnosed based on the incorrect or incomplete data supplied. The safety integrity requirements will identify that this type of hazard must be avoided.

Providing online access to medical records for medical practitioners increases the risk that this highly confidential data may fall into the wrong hands [3] [34] [49]. Further, it should be noted that the medical database has evolved on trust. The patient is willing to their divulge personal and most private details on the basis that this data will not be revealed inappropriately. Failure to comply may put at risk patient consent and participation in new health informatics technologies.

We have already discussed security risks and ensuring the security of full medical records is a similar problem. However, online medical records are also being increasingly made available for medical research [3] [34]. This introduces a need for "inference control", defined by [3] as "how one protects information in the (often lightly) deidentified data ...". There is a risk that it may be possible to reconstruct identity from such data.

5 COTS Components & Risk

In today's economic climate it is no longer cost effective to build entire systems from scratch, and more and more software is being constructed from Commercial Off-The-Shelf (COTS) components. Products can be constructed from already existing components in a much shorter time frame, with possibly fewer staff, and are thus developed at a lower cost. Other motivations for using COTS components include the provision of desired functionality, greater user acceptance by promoting a consistent look and feel, a higher degree of quality assurance as the product has been thoroughly tested by other customers, a greater likelihood of readily available support services and a commercial product is more likely to evolve to take advantage of future hardware. [29] [28]

However, a number of authors have cautioned that the increasing use of COTS components also introduces a new set of possible risks. Components often provide more functionality than is actually required, and these unneeded services may interfere with intended functions [33][29]. As the source code may not be available, it is impossible to check if there is any malicious code such as viruses present [13][29]. In addition, [8] rgues that rapid changes associated with COTS releases and Internet and web-based systems makes it impossible to produce "air-tight" requirements.

Lindqvist and Jonsson [27] suggest that combining COTS and Internet connectivity generates the potential for adverse impacts on the security of the system. Their

list of factors contributing to security risks includes the possibility of bugs in the component, knowledge of the design details by a selected group, not including the customer, and insufficient or incorrect documentation to enable the customer to integrate the component securely in addition to the factors mentioned in the preceding paragraph. They further claim these risks have their origins in the design of the component and are thus beyond the control of the developer using the component.

Both [28]and [32] point out that COTS components are usually designed for other, more generic purposes and are unlikely to have been subjected to the level of verification and validation required for safety critical systems. This affects a large class of applications given that an increasing number of today's applications may be classified as critical.

6 Is this a problem?

Most SE processes and project management processes include risk management, however, SMEs, who make up the majority of today's software developers, do not necessarily follow any standard process. As [41] states "Most software project managers do it (risk analysis) informally or superficially, if they do it all."

Current calls to tailor SE methods for SMEs place little or no focus on how SMEs can handle risk for different situations. Rather than simply describing a method and ways in which the SME can tailor the method, a SE process for SMEs must include a method for assessing risks associated with the tailoring process and the integrity of the software, as well as the usual business risks.

7 Conclusion and Future Work

Today's software projects are facing greater risks than at any other time. This is largely due to much of the software development effort shifting from large organisations to small teams employed largely by SMEs, the trend to constructing software from COTS and the changing character of application domains. Such projects involve three dimensions of risk, generic project risks, risks associated with adapting processes intended for larger teams and risks inherent in the nature of the product.

There is a need for SE processes that can address these elements of risk, particularly for SMEs. We are currently working towards adapting an existing process (the Unified Process) to be both customizable and fully integrate consideration of these risks so that that there is both the opportunity to selectively tailor the process and assess the impact such tailoring may have on the quality of the final product.

8 References

[1] Australian Information Industry Association (2001) Some statistics about the Australian IT&T Industry: Business Size. http://www.aiia.com.au/business_size.html, accessed 27 April, 2001

[2] Ahuja, V. (2000) Building trust in electronic commerce , IT Professional , 2 : 3 , pp 61 –63

[3] Anderson R. J. (2000) Privacy Lessons from Healthcare, IEEE

[4] Beck, Kent (1999) Embracing change with extreme programming. IEEE Computer 32:10, pp70-77

[5] Boddie J. (2000a) Do we ever really scale down? IEEE Software, 17:5, pp79, 81

[6] Boddie J. (2000b) John Responds (to Laitenan, 2000a) IEEE Software, 17:5, p 80

[7] Boehm B. (1989) Software Risk Management. IEEE Computer society Press

[8] Boehm B. (2000) Requirements that Handle IKIWIS, COTS, and Rapid Change. IEEE- Computer, 33:7 pp 99-102

[9] Budgen, D. (1999) Software design methods: life belt or leg iron? IEEE Software, 16:5, pp 133 -136

[10] Catedra del Software (1999) Propuestas para fortalecer la industria del software en Antioquia, Centro de Ciencia y Tecnologia de Antioquia

[11] Cert (2001a) Meet the CERT Coordination Centre. http://www.cert.org/meet_cert/meetcertcc.html accessed 10/5/01.

[12] Cert (2001b) CERT/CC 2000 Annual Report. http://www.cert.org/annual_rpts/cert_rpt_00.html accessed 10/5/01

[13] Devanbu, P.; Fong, P.W.-L.; Stubblebine, S.G. (1998) Techniques for trusted software engineering. Proceedings of the 1998 International Conference on Software Engineering, pp 126 –135

[14] Drobik, Alex (2000) e-business is not easy business. The Computer Bulletin, January 2000. pp 27-29

[15] Fairley, R. (1994) Risk management for software projects. IEEE Software 11:3 pp 57-67

[16] Fayad M. E., Laitinen M. & Ward R. P. (2000) Software Engineering in the Small. Communications of the ACM 43:3 pp 115 - 118

[17] Grigsby, J.; Barton, P.L.(1998) Telecommunications Technology, Health Services and Technology Assessment. Proceedings of the Pacific Medical Technology Symposium, pp 12 - 15

[18] Heard S., Grival T., Schloeffel P. & Doust J. (2000) The benefits and difficulties introducing a national approach to electronic health records in Australia. Repot to the Electronic Records task Force, Commonwealth Dept. of Aged and Health Care, Australia

[19] Industry Science Resources (2001) Issues Paper - Expansion of the CRC Program

[20] Karolak D W (1996) Software engineering risk management. IEEE Computer Society Press

[21] Knauber P., Muthig D., Schmid K. & Wide T. (2000) Applying product line concepts in small and medium-sized companies. IEEE Software, 17:5 pp 88-95

[22] Kornbluh, Ken (2000) Technical Software, IEEE Spectrum, 37:1, pp 58 – 62

[23] Kostenko Constantin (2001) No Title, http://www.kostenko.com/smp_paper2.htm, accessed 5/5/01.

[24] Laitenan M (2000a) Mauri responds (to Boddie, 2000a). IEEE Software, 17:5, p81

[25] Laitenan M (2000a) Scaling down is hard to do. IEEE Software, 17:5, pp 78, 80

[26] Le Grice, Martin (2000) What the developer wants. Software Engineering Australia May 2000 Special Edition pp 30-31

[27] Lindquist U. & Jonsson E. (1998) A Map of Security Risks Associated with Using COTS, IEEE Computer, June 1998, pp 60-66

[28] Lindsy P. & Smith G. (2000) Safety Assurance of Commercial-Off-The-Shelf Software, Accessed 29/6/00, http://www.sea.org.au/seact/sea2000/pres/lin1/paper.htm

[29] Longstaff, T.A.; Chittister, C.; Pethia, R.; Haimes, Y.Y. (2000) Are we forgetting the risks of information technology? Computer , Volume: 33 Issue: 12, pp 43 -51

[30] Maibaum, T. (1993) Taking more of the soft out of software engineering. Proceedings of the Seventh International Workshop on Software Specification and Design, pp 2 –7

[31] Manchala, Daniel W. (2000) E-commerce Trust Metrics and Models. IEEE Internet Computing, march-April 2000, pp 36-38

[32] McDermid, J.A. (1996) COTS: the expensive solution? IEE Colloquium on COTS and Safety Critical Systems (Digest No. 1997/013), pp 1/1 -1/4

[33] McDermid, J.A. (2000) Complexity: concept, causes and control. Proceedings. Sixth IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2000, pp 2 –9

[34] McCandless M. (1998) Staying Healthy in a Wired World, IEEE Intelligent Systems, Jan/Feb 1998, pp 2 -3

[35] Mead, Nancy R., Ellison, Robert, Linge, Richard C., Lipson, Howard F., & McHugh, John (2000) Life-Cycle Models for Survivable Systems. Proceedings of the IEEE 3rd Information Survivability Workshop, ISW-2000.

[36] Moitra D. (1999) Software Engineering in the Small. IEEE Computer, 32:10 pp 39-40

[37] Myerson M. (1996) Risk Management Processes for Software Engineering Models. Artech House

[38] Newsom, D.E.; Herzenberg, C.L.; Swieltik, C.E. (1999) Value of the Internet in emergency response. Communication Jazz: Improvising the New International Communication Culture: Proceedings of the 1999 IEEE International Professional Communication Conference, IPCC 99, pp 35 -40

[39] Pfleeger S. L. (1999) Understanding and improving technology transfer in software engineering. The Journal Of Systems Software, 47 pp 111-124

[40] Platt, A.-B. (1999) The usability risk. Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, pp 396 -400

[41] Pressman Roger S. (2000) Software Engineering: A Practitioner's Approach. (5th Edition) McGraw Hill

[42] Pressman Roger S. (2001) Adaptable Process Model. Hypertext version accessed 4 April 2001. http://www.rspa.com/apm

[43] Roberts Tom (1999) Why Can't we Implement This SDM? IEEE Software, Nov/Dec 1999, pp 70 - 75

[44] Robinson Richard M., Anderson Kevin J. & Meiers Simon (2001) Risk & Reliability - An Introductory Text (3rd Edition, 3rd revison), Risk & Reliability Associates, online version available from www.r2a.com

[45] Ropponen, J.; Lyytinen, K. (2000) Components of software development risk: how to address them? A project manager survey. IEEE Transactions on Software Engineering, 26:2, pp 98 -112

[46] Roy G. G. & Woodings T. L. (2000) A Framework for Risk Analysis in Software Engineering

[47] Sabbah, D. (2001) Software engineering and the Internet. Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, pp 655 -655

[48] Shimeall Timothy J. & McDermott john J. (1999) Software Security in an Internet World: an Executive Summary. IEEE Software, July/August 1999, pp 58-61

[49] Smith E. & Eloff J. (2000) Cognitive Fuzzy Modelling for Enhanced Risk Assessment in a Health Care Institution, IEEE Intelligent Systems, March-April 1998, pp 69-75

[50] SMT Consulting P/L (1998) Stocktake of Australia's Information Industries: Strengths weaknesses, opportunities and threats. The Information Industries and Online Taskforce, Department of Industry, Science and Tourism.

[51] Sommerville 1. (2001) Software Engineering. (6th Edition) Addison Wesley

[52] Tran, V.N.; Lin, D.-B. (1999)Application of CBSE to projects with evolving requirements-a lesson-learned. Proceedingsof the Sixth Asia Pacific Software Engineering Conference, (APSEC '99) pp 28 -37

[53] Trussell, L. (1999) Essential software development methodology. IEEE Power Engineering Society 1999 Winter Meeting, Volume: 1 , pp 357 -361

[54] Van Scoy, R (1992) Software Development Risk: Opportunity, Not Problem CMU/SEI-92-TR-30

[55] Yan, G. & Paradi, J.C. (1999) Internet-the future delivery channel for banking services? Proceedings of the Thirty-First Hawaii International Conference on System Sciences, Volume: 4, pp 290 -299

[56] Yan, G.; Paradi, J.C. (1999) Success criteria for financial institutions in electronic commerce. Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, 1999. HICSS-32.

[57] Zakon Robert H. (2001) Hobbes' Internet Timeline v5.3. http://www.zakon.org/robert/Internet/timeline/, accessed 10/05/01