2002

Why don't we teach software engineers about the law?

Anne Fuller
University of Wollongong, alf876@uow.edu.au

Peter Croll
University of Wollongong

Why don't we teach software engineers about the law?

Abstract

Much work has been done in recent years developing software engineering curricula. SE research has traditionally focused on the needs of very large corporations undertaking equally mammoth and complex development projects, consequently, current curricula tend to focus on this model. Yet by far the majority of software development is undertaken by Small to Medium Enterprises. The rise of the internet as a platform for commercial applications has partly driven this move away from monolithic software development. Depending on the nature of the application itself many of these products can be described as 'critical' with the failure of such a product more likely to involve substantial losses for the customer. Many of these smaller 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. One of the newer elements of risk includes the possibility of litigation. While current curricula already include elements of ethics and social responsibility, the changing nature of both development teams and today's software products places today's software engineer in a position where litigation is a very real possibility. In this paper we discuss the factors contributing to the possibility of litigation in detail, and suggest that consideration of legal consequences of decision-making should be included as a vital part of teaching software engineers about risk management.

Keywords

about, engineers, software, law, teach, why, we, t, don

Disciplines

Engineering | Science and Technology Studies

Publication Details

Fuller, A. & Croll, P. (2002). Why don't we teach software engineers about the law?. Journal of Law and Information Science, 12 (1), 139-153.

Why don't we teach Software Engineers about the law?

ANNE FULLER, PETER CROLL*

Abstract

Much work has been done in recent years developing software engineering curricula. SE research has traditionally focused on the needs of very large corporations undertaking equally mammoth and complex development projects, consequently, current curricula tend to focus on this model.

Yet by far the majority of software development is undertaken by Small to Medium Enterprises. The rise of the internet as a platform for commercial applications has partly driven this move away from monolithic software development. Depending on the nature of the application itself many of these products can be described as "critical" with the failure of such a product more likely to involve substantial losses for the customer.

Many of these smaller 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.

One of the newer elements of risk includes the possibility of litigation. While current curricula already include elements of ethics and social responsibility, the changing nature of both development teams and today's software products places today's software engineer in a position where litigation is a very real possibility.

In this paper we discuss the factors contributing to the possibility of litigation in detail, and suggest that consideration of legal consequences of decision-making should be included as a vital part of teaching software engineers about risk management.

1. Introduction

For many years after the birth of Software Engineering (SE) in 1968, software development was generally concerned with the production of custom software under contract for large corporations. 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

* University of Wollongong {annef, croll}@uow.edu.au

risks associated with shortcuts and mistakes and ensure the quality of the software product.[1] Although SE research has produced many such processes and supporting tools, these have mainly benefited those few companies large enough to take advantage of these advances.[2]

However, 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. The escalation in internet based products has brought with it a corresponding 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. Combining these factors, increases the likelihood that the integrity of these products may be compromised.[3]

In this paper we discuss the changing nature of today's software products, and describe particular areas where the developer is now even more exposed to the likelihood of litigation. We then discuss shortcomings in SE curricula leaving software engineers ill-prepared for the realities of today's litigious environment. Finally, we briefly describe how such legal considerations should be included in the SE curriculum.

2. SME's and Risk

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, Fayad *et al*[4] 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

---

1 Roberts, T., 'Why Can't We Implement This SDM?', *IEEE Software*, Nov/Dec 1999, pp 70-75.

2 Moitra, D., 'Software Engineering in the Small', *IEEE Computer*, vol. 32, no. 10, 1999, pp 39-40.

3 Fuller, A., Croll, P., & Garcia, O., 'Why software engineering is riskier than ever', APAQS 2001, Proceedings of the 2nd Asia Pacific Conference on Quality Software, IEEE Computer Society.

4 Fayad, M.E., Laitinen, M. and Ward, R.P., 'Software Engineering in the Small', *Communications of the ACM*, vol. 43, no. 3, pp 115-118.

company size, development mode, development size and development speed. We have previously presented evidence that this situation is similar in other countries, including Australia.[5]

Thus the 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 the nature of the product.

3. Software Risks

There is a growing dependence on computers for business and life-critical functions. Thus it is essential that such applications offer some guarantee of integrity and that risk techniques formerly confined to safety critical systems are now more broadly applied.[6] 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.

3.1 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.[7]

In November 1988, the Software Engineering Institute set up a Computer Emergency Response Team Coordination Center (CERT/CC), to coordinate communication among experts during security emergencies.[8] The CERT/CC 2000 Annual Report states that

---

5 Fuller, A., Croll, P., Awyzio, G. and Garcia, O., 'Re-emphasising risk in the Software Engineering Syllabus', *Proceedings of the Fifth IASTED International Conference on Software Engineering and Applications (SEA 2001)*, August 21-24, 2001, Anaheim, US; Fuller, A., Croll, P. and Garcia, O., 'Why software engineering is riskier than ever', presented at the 2nd Asia Pacific Conference on Quality Software (APAQS 2001), 10-11 December, 2001, Hong Kong.

6 McDermid, J.A., 'Complexity: concept, causes and control', *Proceedings of the Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000)*, pp 2-9.

7 Shimeall, T.J. and McDermott, J.J., 'Software Security in an Internet World: An Executive Summary', *IEEE Software*, July/August 1999, pp 58-61.

8 CERT, Meet the CERT Coordination Centre', 2001,

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

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.[10] Many start-up 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.[11] 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.[12]

Newsom *et al*[13] 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.

---

http://www.cert.org/meet_cert/meetcertcc.html, accessed 10 May, 2001; Zakon, R.H., 'Hobbes' Internet Timeline', vol. 5.3, http://www.zakon.org/robert/Internet/timeline/, accessed 10 May, 2001.

9 CERT, CERT/CC 2000 Annual Report, http://www.cert.org/annual_rpts/cert_rpt_00.html, accessed 10 May, 2001.

10 Shimeall and McDermott, *supra* n.7.

11 Platt, A.B., 'The useability risk', *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 1999, pp 396-400.

12 Ibid; Kornbluh, K., 'Technical Software', *IEEE Spectrum*, vol. 37, no. 1, 2000, pp 58-62.

13 Newsom, D.E., Herzenberg, C.L. and Swieltik, C.E., '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.

3.2 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",[14] to "the integration of internal and business processes through information and communication technologies".[15] 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.[16] These authors, together with Yan & Paradi[17] have suggested a number of ingredients vital for building mutual trust.

However there is still a lack of e-commerce standards accepted by either businesses and consumers.[18] 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.

3.3 Health

Medical databases are proliferating, creating a potentially hazardous environment.[19] With the safety of patients at risk it becomes even

---

14 Drobik, A., 'e-business is not easy business', *The Computer Bulletin*, January 2000, pp 27-29.

15 Ahuja, V., 'Building trust in electronic commerce', *IT Professional*, vol. 2, no. 3, 2000, pp 61-63.

16 Ibid; Manchala, D.W., 'E-commerce Trust Metrics and Models', *IEEE Internet Computing*, March-April 2000, pp 36-38.

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

18 Ibid.

19 Heard, S., Grival, T., Schloeffel, P. and Doust, J., *The benefits and difficulties introducing a national approach to electronic health records in Australia*, Report to the Electronic Records Taskforce, Commonwealth

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.[20] Further, it should be noted that the medical database has evolved on trust. Patients are willing to divulge their 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.[21] This introduces a need for "inference control", defined by Anderson[22] 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.

3.4 COTS Components

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

---

Department of Aged Care and Health, Australia Industry Science Resources Issues Paper, 2001.

20 McCandless, M., 'Staying Healthy in a Wired World', *IEEE Intelligent Systems*, Jan/Feb 1998, pp 2-3; Anderson, R.J., 'Privacy Lessons from Healthcare', *IEEE*, 2000; Smith, E. and Eloff, J., 'Cognitive Fuzzy Modelling for Enhanced Risk Assessment in a Health Care Institution', *IEEE Intelligent Systems*, March-April 2000, pp 69-75.

21 McCandless, *supra* n.20; Anderson, *supra* n.20.

22 Anderson, *supra* n.20.

developed at a lower cost. Longstaff *et al*[23] and Lindsay & Smith[24] list a number of other other motivations for using COTS components

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.[25] As the source code may not be available, it is impossible to check if there is any malicious code such as viruses present.[26] Boehm[27] argues that rapid changes associated with COTS releases and internet and web-based systems makes it impossible to produce "air-tight" requirements.

Perhaps of greater concern is the suggestion that combining COTS and internet connectivity generates the potential for adverse impacts on the security of the system.[28] In addition, both Lindsay & Smith[29] and McDermid[30] 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.

---

23 Longstaff, T.A., Chittister, C., Pethia, R. and Haimes, Y.Y., 'Are we forgetting the risks of information technology?', *Computer*, vol. 33, iss. 12, pp 43-51.

24 Lindsey, P. and Smith, G., 'Safety Assurance of Commercial Off-The-Shelf Software', 2000, http://www.sea.org.au/seact/sea2000/pres/lin1/paper.htm, accessed 29 June, 2000.

25 McDermid, *supra* n.7; Longstaff et al, *supra* n.24.

26 Longstaff et al, *supra* n.24; Devanbu, P., Fong, P.W.L. and Stubblebine, S.G., 'Techniques for trusted software engineering', *Proceedings of the 1998 International Conference on Software Engineering*, pp 126-135.

27 Boehm, B., 'Requirements that Handle IKIWIS, COTS, and Rapid Change', *IEEE Computer*, vol. 33, no. 7, 2000, pp 99-102.

28 Lindquist, U. and Jonsson, E., 'A Map of Security Risks Associated with Using COTS', *IEEE Computer*, June 1998, pp 60-66.

29 Lindsey and Smith, *supra* n.24.

30 McDermid, J.A., 'COTS: the expensive solution?', *IEEE Colloquium on COTS and Safety Critical Systems*, Digest No. 1997/013, pp 1/1-1/4.

4. Why is this a problem?

Since 1995 there has been an escalation in computer related litigation in the US[31] and it has been estimated that the cost of litigation is actually rising faster than any other cost of software development.[32] It surely cannot be too long before the situation is similar in Australia.

US software suppliers commonly limit their exposure by including a limitation of liability clause in their contracts. Such clauses limit the suppliers' liability to amounts paid under the contract and claims for damages over and beyond that are generally being disallowed. For example, suppose a piece of software is developed using third party software libraries for which the developer fails to pay the appropriate license fees. The copyright owner may sue the unsuspecting user of the software for damages, however the user cannot recover any more from the supplier than was previously paid under the contract.[33]

In Australia, however, the *Trade Practices Act* and *Fair Trading Acts* provides a number of implied conditions that cannot be excluded from any contract.[34] One such condition is that the supplier has the right to sell the goods and they are free from any charge or encumbrance. For goods and services supplied in the course of business those goods must be of "merchantable quality" and "reasonably fit for the purpose", while services "will be rendered with due care and skill".

Thus Australian developers cannot limit their liability to the same extent as in the US. Actions have been settled in Australian courts where the damages recovered include compensation for loss of opportunity and expenses incurred remedying the problem, in addition to recovery of payments under contract.[35] An organization that faces losses when the security of a web-based application is

---

31 Cosgrove, J., 'Software Engineering and the Law', *IEEE Software*, May/June 2001, pp 14-16.

32 DeMarco, T. and Lister, T., 'Both Sides Always Lose: Litigation of Software Intensive Contracts', *Crosstalk*, vol. 13, no. 2, February 2000, pp 4-6.

33 Grossman, M., '"Limitation of Liability" Clauses in High-Tech Contracts', 2001, GigaLaw.com, http://www.gigalaw.com/articles/2001/grossman-2001-06-p1.html, accessed 9 June, 2001.

34 White, S., 'Trade Practices Act – Implied Warranties', 1996, www.computerlaw.com.au/impwarr.html, accessed 9 June, 2001.

35 Jones Condon, 'Recovering $2 Million Loss for Faulty Software', 1999, http://www.jonescondon.com.au/news/survive_sept99/surv_loss.html, accessed 9 June 2001.

compromised or due to a defective e-commerce application, may seek some recompense from the developer. A possible defense might be having a record of the system development record process showing that all reasonable steps have been taken.[36] However, the selective practice of recommended SE procedures increases the exposure of the developer to the charge of not having rendered the service with "due care and skill".

While it is possible for an organization to take out product liability insurance, such insurance does not necessarily cover computer software and the data it organizes.[37] Rigney also warns that software failures that seriously corrupt stored data are "a potential liability issue for the software manufacturer and the supplier."

Underwood[38] reports that software developers are also legally liable for the safe operation of their products. The Australian *Trade Practices Act* includes the intent of the 1985 European Community Product Liability Directive. According to Article 6 of this directive, a product is defective "… when it does not provide the safety which a person is entitled to expect …'. Underwood[39] goes on to state that an acceptable defense in those countries that have adopted this concept is based on the premise that the defect could not have been discovered using scientific and technical knowledge available at the time. For example car manufactures undertake extensive research and testing to validate designs before product release, and can defend themselves by citing this process.[40]

However, as previously discussed, SMEs, who make up the majority of today's software developers, do not necessarily follow any standard process. In addition, according to Rowland & Rowland[41], there is a "significant proportion" of developers involved in the

---

36 Cosgrove, *supra* n.31.

37 Rigney, K., 'Software and Data Damage Not Covered', Media Release, Insurance Financial Services Division, Phillips Fox, 1998 http://www.phillipsfox.com.au/general/N0000062.htm, accessed 11 June, 2001.

38 Underwood, A., *Software Engineering Australia: Certification of Software Engineers*, report presented to a joint workshop held by Software Engineering Australia and the Software Quality Association of South Australia, May 1999. Available from www.sqa.asn.au/presentations/may99/underwood.pdf, accessed July 22, 2001.

39 Ibid.

40 Cosgrove, *supra* n.31.

41 Rowland, J.J. and Rowland, D., 'Professional competence in safety-related engineering', *Software Engineering Journal*, March 1999, pp 43-48.

production of safety-related systems who are not aware that such systems involve "special problems". Thus many of today's e-health applications are being developed with little in the way of standard SE practices, let alone "state of the art". How long can it before the software will be blamed for human injury or even death, and the developers deemed negligent if unable to show adherence to accepted standards for development of safety critical software?

5. Why Teach Legal issues to Software Engineers?

In the past when the graduate software engineer would most likely work for a large development company or perhaps the in-house IT section of a government department the issue of developer liability was not a concern; any threat of or actual litigation would be handled by the legal department and senior management. The software engineer, busy with new projects, would be unlikely to be involved and may even remain ignorant of any such action.

The rise of the SME software development house has brought with it a newer organisational model for a software development team. More and more teams are being "constructed" from individual consultants, working for themselves. Today's SE graduates may spend as little as 1 – 2 years as an employee before branching out on their own. Our current syllabi do not prepare SEs for this environment.

There have been a number of proposals in recent years for teaching some law, along with social and ethical issues to students of both Software Engineering and Computer Science (CS).[42] The former two proposals originate in the US and focus heavily on Ethics and Society, with some reference to copyright and privacy issues. Cifuentes and Fitzgerald discuss the introduction of a legal strand into the CS curriculum at the University of Queensland. This strand covers topics such as IP, software licensing agreements and contracts, as well as optional units looking at a number of cyberlaw issues and the digital society.

None of these however, appear to examine the possibility of liability on the part of the developer. Given that US contract law permits the

---

42 Martin, C.D., Huff, C., Gotterbarn, D. and Miller, K., 'Introducing a Tenth Strand in the CS Curriculum', *Communications of the ACM*, 39/12, pp 75-84; Sedlet, S., 'Computers, Ethics, Law and Society: What do we teach undergraduates?', *Proceedings of the 1999 International Symposium on Technology and Society, Women and Technology: Historical, Societal and Professional Perspectives*, pp 249-253; Cifuentes, C. and Fitzgerald, A., 'Introducing a Legal Strand in the Computer Science Curriculum', *Proceedings of the Third Australasian Conference on Computer Science Education*, 1998, pp 19-26.

inclusion of limitation of liability clauses it is perhaps not so surprising that this does not appear in the US proposals. However, the Australian strand does not appear to cover this aspect either.

We have reported elsewhere our belief that risk can and should be given status as a complete subject in its own right and that the consideration of risk be integrated across all phases of the SE curriculum, instead of playing a minor role as a sub-component of Software project management.[43] The risk of legal liability for failure of a software product, is just one aspect of risk which needs to be re-emphasised in the SE curriculum along with consideration of generic project risks, risks associated with adapting processes intended for larger teams and risks inherent in the nature of the product.

6. Conclusion

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 trends have introduced a number of risks not previously associated with software projects. One such risk which has been mostly overlooked, is the possibility of litigation where a defective product has lead to economic loss or personal injury

The software engineer working on large monolithic development efforts might never be aware that such litigation is taking place, being shielded from the knowledge by the many organisational layers between the software development project team and the legal department. However, the smaller, leaner software houses of today provide no such protective layer. Indeed, the rise of small contractual teams, which may comprise as little as one or two software engineers, opens the possibility for a software engineer to be personally held liable for failure of a product. By failing to educate our SE students about the possibility of litigation we are failing in our own duty to ensure these students are prepared to face all the realities of today's SE environment.

---

43 Fuller, Croll, Awyzio and Garcia, 'Re-emphasising risk in the Software Engineering Syllabus', *Proceedings of the Fifth IASTED International copnference on Software Engineering and Applications* (SEA 2001) August 21-4, Anaheim, US.

Bibliography

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

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

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

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

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

Cifuentes C & Fitzgerald A (1998) Introducing a Legal Strand in the Computer Science Curriculum, Proceedings of the third Australasian conference on Computer science education, pp 19 - 26

Cosgrove, John (2001) Software Engineering and the Law, IEEE Software, May/June 2001, pp 14 – 16.

DeMarco T & Lister T. (2000) Both Sides Always Lose: Litigation of Software intensive Contracts. Crosstalk 13/2, Feb 2000, pp 4 - 6

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

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

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

Fuller A, Croll P., Awyzio G. & Garcia O, Re-emphasising risk in the Software Engineering Syllabus, Proceedings of the Fifth IASTED International Conference on Software Engineering and Applications (SEA 2001), August 21-24, 2001 Anaheim, USA

Fuller, A., Cross, P., & Garcia, O., (2001) 'Why software engineering is riskier than ever', APAZS 2001, Proceedings of the 2nd Asia Pacific Conference on Quality Software', IEEE Computer Society.

Grossman Mark (2001) “Limitation of Liability” Clauses in High-Tech Contracts, GigaLaw.com,

http://www.gigalaw.com/articles/2001/grossman-2001-06-p1.html, accessed 9 June 2001

Heard S., Grival T., Schloeffel P. & Doust J. (2000) The benefits and difficulties introducing a national approach to electronic health records in Australia. Report to the Electronic Records task Force, Commonwealth Dept. of Aged and Health Care, Australia Industry Science Resources (2001) Issues Paper - Expansion of the CRC Program

Jones Condon (1999) Recovering $2 Million Loss for Faulty Software, Accessed 9 June 2001, http://www.jonescondon.com.au/news/survive_sept99/surv_loss.html

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

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

Lindsey 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

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

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

Martin C D, Huff C, Gotterbarn D & Miller K (1996) Introducing a tenth Strand in the CS curriculum, Communications of the ACM, 39/12, pp 75 - 84

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

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

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

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

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

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

Rigney K. (1998) Software and Data Damage Not Covered. Media Release from Insurance Financial Services Division, Phillips Fox, http://www.phillipsfox.com.au/general/N0000062.htm, Accessed 11 June, 2001

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

Rowland J J & Rowland D (1995) Professional competence in safety-related engineering. Software Engineering Journal, March 1995, pp 43 - 48

Sedlet S (1999) Computers, Ethics, law and Society: What do we teach undergraduates? Proceedings of 1999 International Symposium on Technology and Society, Women and Technology: Historical, Societal, and Professional Perspectives, pp 249 - 253

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

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

Underwood A (1999) Software Engineering Australia: Certification of Software Engineers. Report presented to a joint workshop held by Software Engineering Australia and the Software Quality Association of South Australia, May 1999. Available from www.sqa.asn.au/presentations/may99/underwood.pdf, accessed July 22, 2001

White S. (1996) Trade Practices Act – Implied Warranties. www.computerlaw.com.au/impwarr.html, Accessed 9 June 2001

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.

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