February 2002

A new approach to teaching software risk management with case studies

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

P. Croll
University of Wollongong

L. Dei
University of Wollongong

A new approach to teaching software risk management with case studies

Abstract

Software development is inherently risky, however the need to adapt processes intended for larger organisations introduces a new element of risk. In addition, the nature of many new software products can be described as "critical" and therefore should undergo a formal risk assessment procedure. Despite the majority of software projects involving these additional elements of risk, risk management planning is virtually non-existent, as managers have not been trained in risk management. Few current software engineering curricula provide comprehensive coverage of risk, nor any practical experience in risk assessment. In this paper we propose that risk be repositioned in the software engineering core body of knowledge, and discuss preliminary requirements for a one semester course on risk. As experience is considered an essential element of successful risk assessment, a feature of the course is the use of case studies based on real projects to simulate an historical perspective for students.

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Fuller, A, Croll P & Dei, L, A new approach to teaching software risk management with case studies, Proceedings 15th Conference on Software Engineering Education and Training, 25-27 February 2002, 215-222.

A New Approach to Teaching Software Risk Management with Case Studies

Anne Fuller, Peter Croll, Limei Di
University of Wollongong
{annef, croll, limei}@uow.edu.au

Abstract

Software Engineering 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. Software development is inherently risky, however the need to adapt processes intended for larger organisations introduces a new element of risk. In addition, the nature of many new software products can be described as "critical" and therefore should undergo a formal risk assessment procedure.
Despite the majority of software projects involving these additional elements of risk, risk management planning is virtually non-existent, as managers have not been trained in risk management. Few current software engineering curricula provide comprehensive coverage of risk, nor any practical experience in risk assessment. In this paper we propose that risk be repositioned in the software engineering core body of knowledge, and discuss preliminary requirements for a one semester course on risk. As experience is considered an essential element of successful risk assessment, a feature of the course is the use of case studies based on real projects to simulate an historical perspective for students.

1. Introduction

For the more than 30 years since its birth, Software Engineering (SE), has been 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 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].
Yet most of today's software is being developed by Small to Medium Enterprises (SMEs) rather than large companies. Fewer than 6% of all software houses in the United States have more than 50 employees [3] and we have previously shown that this situation is similar in other countries, including Australia [4][5]. 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 [3]. The advent of methodologies such as Extreme Programming and adaptable process models such as that described by [6] are attempts to redress this situation. However, these approaches do not

Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02)
1093-0175/02 $17.00 © 2002 **IEEE**

give any guidance for decision makers with regard to the risk associated with any tailoring of the process [4].

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 risk. We then discuss shortcomings in SE curricula leaving software engineers ill-prepared to manage the risks facing today's software projects.

One of the reasons advanced for lack of coverage of risk management at the undergraduate level is the notion that risks cannot be adequately assessed without prior experience. We do not dispute the importance of having some historical basis for an assessment. Rather, the course we propose overcomes this by using real and relevant case studies that allow the students to compare their assessments against actual outcomes. Effectively the students build up their project "experience" as the course progresses.

2. 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 applicable [7]. 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 reliable and secure [8][9][10]. However, there is a lack of e-commerce standards accepted by either businesses or consumers [11] and a growing concern that in the rush to get a product online, quality may be compromised [12][13].

The medical profession's growing reliance on electronic medical databases not only requires security guarantees [14] [15][16] but introduces safety hazards as well [17]. 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.

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. 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 [7][18]. As the source code may not be available, it is impossible to check if there is any malicious code such as viruses present [18][19]. Boehm [20] argues that rapid changes associated with COTS releases and internet and web-based systems makes it impossible to produce "air-tight" requirements, while [21] suggests that combining COTS and internet connectivity generates the potential for adverse impacts on the security of the system. In addition, both Lindsay & Smith [22] and McDermid [23] 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.

Thus 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 today's software product.

3. Risk in the Current Curriculum

The IEEE Computer Society and the Association for Computing Machinery co-operative effort to establish a Software Engineering Body of Knowledge (SWEBOK) aims, among other objectives, to "characterize the contents of the software engineering discipline" and "provide a foundation for curriculum development …". SWEBOK proposes Software Engineering Management as a core component of the curriculum and includes risk management as one element of this knowledge area. [24]

Direct linkages can be made between the simplified spiral model and the IEEE/ACM topic areas with the notable exception of risk analysis [6]. Despite its perceived importance in an accepted SE model, (the spiral model), risk is not afforded a corresponding place among topic areas for SE education, but instead is buried inside project management. This neglect of the significance of risk pervades SE education.

Standard software engineering texts such as Sommerville [25] and Pressman [26] provide minimal coverage of risk. Pressman includes a chapter on risk, but does not show how risk can be integrated across the entire process. Somerville concentrates on risk analysis in terms of safety critical systems, and also includes a brief introduction to risk management as part of his discussion of the Spiral Model of the software process. Jacobsen [27] discusses risk in terms of moving from some other process to his Object Oriented Software Engineering process.

MacDonell and Gray [28] also imply that risk management enjoys limited treatment in standard SE texts by suggesting the use of [29] to "provide a more general and comprehensive text on risk management …". Karolak [29] cites both the failure by the university education system to adequately address risk management for software projects and the view that risk management should be integrated into the entire software development lifecycle among the motivations for his work.

With so little significance given to risk management in current SE syllabi, is it any wonder that today's software engineers fail to make more than a casual pass at this task.

4. Re-emphasising risk in the syllabus

Bagert [30] provide comprehensive guidelines for development of software engineering programs. Risk analysis is included as part of the Software Management Component of their suggested core knowledge areas. (The body of knowledge on which their proposed curriculum is predicated was developed independently of the work of the SWEBOK committee.) The curriculum comprises nine modules with an "introduction to project planning and risk management" one topic out of a list of 13 topics comprising one of these modules. Risk analysis is also listed in the content of the Module the authors title "Senior Design Project".

A recent updating of the description of SWEBOK'S software engineering management knowledge area presents an alternative breakdown of topics, separating generic management issues from the more specific software engineering management and places risk management firmly in the area of generic management, quite distinct from software engineering management [28].

This insistence that risk be part of management has contributed to its inadequate coverage in most SE curricula. In a one semester course, comprising 13 or 14 weeks, management per se is often relegated to 1 week or less, with risk management a minor component. As we have shown, today's software projects are facing increased risk exposure, yet little attention is paid

to risk management, particularly on the large majority of projects being undertaken by SMEs. The main reason for this is that "there is little to instruct software project managers on how to handle risk …" [31]. Even the Spiral Model, which recognizes that risk plays a vital role in the SE process, uses "risk analysis" as a generic term, giving no specifics regarding what is/should be part of the process [29] [32]. The obvious solution, therefore, is to ensure risk is given more appropriate coverage when educating potential software engineers.

Consider again the three dimensions of risk exposure for SME software developers: business, process and integrity. A risk management curriculum for educating software engineers to handle these risks should include among its expected outcomes the demonstrated ability to:

1. identify risks (business, process and integrity)
2. calculate risk probabilities for quantitative assessments and to set realistic bands for qualitative assessments
3. calculate quantitative and determine qualitative risk impacts
4. determine when applying qualitative versus quantitative assessment is appropriate
5. perform safety and hazard analysis of a software product
6. prepare and carry out risk mitigation, monitoring and management strategies.

It is imperative that risk be considered throughout the software process and not just as one minor aspect of project management. Any part of the process may be risk driven. For example, Boehm [20] advocates a risk-driven approach to deciding what should and shouldn't form part of a requirements specification and argues that rapid changes associated with COTS releases and internet and web-based systems makes it impossible to produce "air-tight" requirements. Case studies should be used to demonstrate the assessment of risk factors arising at various stages and their impact on the project. Most importantly, students need the opportunity to apply the methods to identifying, analysing and tracking risks associated with their own software projects.

A number of authors have presented checklists to facilitate the identification and classification of risks [26][33][34]. However, these are mainly concerned with managing those risks we have classified as project or business risks and do not examine risks associated with deployment of the finished product i.e. integrity risks. This has formerly been the realm of safety critical systems, but, as we have shown, more and more software products these days can, and should, be classified as "critical". It is therefore vital that students be familiar with methods for determining the safety integrity level of a given product.

4.1. The need for experience

In the world of safety critical systems, risks can be categorised in terms of **Safety Integrity Levels (SIL)** (i.e. a SIL is a measure of the amount of risk reduction required for a safety-related system to an acceptable/tolerable level of risk). Table 1 shows SIL 1 to 4 and their associated target failure rates.

Safety Integrity Level	**Demand Mode Of Operation** (Probability of failure to perform its design function on demand)
1	$\geq 10^{-2}$ to $<10^{-1}$
2	$\geq 10^{-3}$ to $<10^{-2}$
3	$\geq 10^{-4}$ to $<10^{-3}$
4	$\geq 10^{-5}$ to $<10^{-4}$

Table 1. Safety Integrity Levels: Target Failure Measures
(taken from IEC 1508 Part 1, shows the target failure rates and related safety integrity levels)

For a given application a SIL can be derived using either a qualitative (i.e. graphs and look-up tables) or quantitative measures. SIL 1 is the lowest risk while SIL 4 is the highest. Examples of low SIL applications might include a machine control system where there is only a remote possibility that a single person receives a negligible injury. A high SIL example would be a power station or weapons system that has the potential to kill hundreds of people in a single incident. These are generalisations since the actual SIL will always have to be calculated for any given application, its location and usage.

The traditional assessment methods for risk and hazard analysis [35][36][37] all rely on people making judgments based on their experience. For safety systems a detailed knowledge of what can go wrong is an essential prerequisite to any meaningful predictions regarding the cause and effects of systems failures. Petroski takes this argument further by stating that teaching history of engineering failures should be a core requirement in any engineering syllabus and take the same importance as the teaching of modern technology [35]. At present, the emphasis on technology is evident in all realms of engineering in particular with the advances in computing. Without an understanding of history or direct experience for a given application then more is unknown and hence the risks are higher [36]. This is demonstrated in Figure 1 where the need for both experience and skills is highlighted. The rationale is that it is unacceptable to be developing a high integrity level system unless the teams involved have both sufficient technical skills and experience.

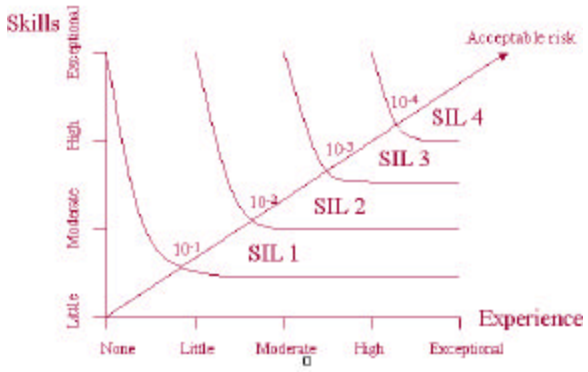

Figure 1. Skills vs Experience Requirements for SIL projects

4.2. Bringing experience into the classroom

Clearly the importance of relevant experience to successful risk assessment is a problem for undergraduate teaching. Many students lack such experience and it can easily become a paper exercise of little direct relevance to them at this stage. It is necessary to develop a mechanism whereby this lack of experience can be addressed and the lessons learned can be retained by the students.

Stratton [37] have shown that relating real world experiences is the most effective. We propose the use of case studies, based on the history of real projects. The case studies will be drawn from industry and students will be asked to perform risk assessments based on data that was available at certain times throughout the project. The students' assessments can then be compared with actual outcomes. In this way the student constructs their own experiential background, becoming progressively more familiar with all kinds of risks and their impacts on particular types of projects.

Three such case studies that might be considered by students in the course are discussed below. All are (unfortunately) synopses of real situations.

Case 1: A project is estimated to be finished in one year, using two programmers. However, due to a change in circumstances, it is desirable that the project be speeded up. A proposal is put forward to increase the number of programmers to six, thus allowing the project to be completed in 4 months. Students are asked to estimate the risks involved in either of these two scenarios. After estimating the above, what is the risk of losing the contract if the company cannot finish the project in 4 month? Ultimately the risk boils down to a choice between producing a more professional product with the possibility of losing the contract or, a guaranteed contract under conditions that may adversely affect the quality of the product and thus cost more in the long run.

Case 2: A company has developed a very comprehensive software product for a particular target industry. However, most of the users in the industry cannot afford the costs of deploying the product. The software is too complex for simple users to handle, and potential clients must dedicate a team of people with knowledge of the industry and advanced computer skills if they are to successfully adopt the package. Thus the developers need to scale down the original product in order to increase market share. They are considering two options: to block out some modules in the original system and thus disable some functionality, or to write another small system from scratch, reusing some code from the old system.

The first option retains the stable structure of the original system, but results in a larger system than necessary. Such a system might be slow and will occupy an unnecessarily large amount disk space, but the risk of bugs creeping into the system appears minimal. The second choice would provide a more usable system, but, as it is essentially a new software product, there is a higher risk of production errors.

Again students are to estimate the risks involved with each option. Issues to be considered include whether either method is more likely to compromise the quality of the final product and the cost of the effort involved in coding, setup, training and after sales support.

Case 3: A local area public health service collects data from patient visits to a number of hospitals and community health centres. Currently the data recorded is used only for statistical purposes. The hospitals and community health services issue different Medical Reference Numbers (MRNs), thus there is no unique identifier for patients. While this may slightly skew statistical results, it does not, of itself, pose a danger to the well being of patients. The health service wishes to use the current system as a basis for establishing a Case History database. The database will provide both hospital and community health staff with access to patient case histories, thus allowing decisions to be made immediately without waiting for detailed medical records to be transferred. The health service intends to migrate all existing data into the new Case History database system. Once the new system is deployed, it is intended that all patients receiving treatment will be issued with a unique MRN to be used at any centre or hospital in the system.

Students are asked to assess the safety integrity level of this system. There are a number of safety risks inherent in the proposed system. Examples include the risk that crucial data from the preexisting system is mis-identified, and thus cannot be matched to the correct patient or a patient under the new system is issued a new MRN and their previous history is not linked correctly. In both these situations a mis-diagnosis may occur or incorrect treatment undertaken, as the full history is not available.

5. Conclusion

Instead of playing a minor role as a sub-component of software project management, risk can and should be given status as a complete subject in its own right and integrated across all phases of the SE curriculum.

The need to re-emphasise risk is largely due to much of the software development effort shifting from large organisations to small teams employed largely by SMEs. 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, however, risk management is at best cursory, at worst non-existent, mostly due to lack of risk management education. This deficiency is often attributed to the notion that risks cannot be adequately assessed without prior experience. The course we have proposed overcomes this by using real and relevant case studies that allow the students to compare their assessments against actual outcomes. Effectively the students develop their own "experience" as the course progresses.

6. References

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

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

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

[4] Fuller A., Croll P. & Garcia O. (2001) Why Software engineering is riskier than ever. To be presented at the 2nd Asia Pacific Conference on Quality Software (APAQS 2001), Hong Kong , Dec 10 –11, 2001

[5] 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)

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

[7] 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

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

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

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

[11] 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.

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

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

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

[15] Anderson R. J. (2000) Privacy Lessons from Healthcare, S&P 2000. Proceedings of 2000 IEEE Symposium on Security and Privacy pp. 78 -79

[16] 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

[17] 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

[18] 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

[19] 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

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

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

[22] 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

[23] 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

[24] Bourque P., Depuis R., Abran A., Moore J. W. & Tripp L. (1999) The Guide to the Software Engineering Body of Knowledge, IEEE Software Vol 16 No 6 35-44

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

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

[27] Jacobsen Ivar (1992) Object Oriented Software Engineering: A Use Case Driven Approach, Addison Wesley

[28] Macdonell S. G. & Gray A. R. (1999) SWEBOK: Software engineering management knowledge area description version 0.6 SWEBOK, www.swebok.org

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

[30] Bagert D., Hilburn T. B., Hislop G., Lutz M., MeCracken M. & Mengel S. (1999) Guidelines for Software Engineering Education Version 1.0, CMU/SEI-99-TR-032, Carnegie-Mellon University/Software Engineering Institute, http://www.sei.emu.edu/topics/collaborating/ed/workgroup-ed.html

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

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

[33] SEI (1993) "Taxonomy -Based Risk Identification" Software Engineering Institute, CMU/SEI-93-TR-6

[34] Collofello J. (2000) "Question List for Software Risk Identification in the classroom', Arizona State University Software Risk Management Home Page

[35] Petroski Henry (1994) "Design Paradigms: Case Histories of error and judgment in Engineering", Cambridge University Press

[36] Croll PR, Chambers C, Bowell M and Chung PWH (1997) Towards Safer Industrial Computer Control Systems, SAFECOM'97, York Univ. 1, pp321-331.

[37] Stratton, M Holcombe & PR Croll, Improving the Quality of Software Engineering Courses through University based Industrial Projects, *Project'9*8, HEFCE workshop on the projects in the Computing Curriculum, Univ. of Sheffield, April 1998, pp145-158.