

A Model for Investigating Software Accidents

Tom McBride

University of Technology, Sydney
mcbride@it.uts.edu.au

A software accident is an unforeseen outcome that arises from a failure of a software project or software product. Death, severe injury or severe financial loss, could arise from such a failure. This paper asserts that there is not yet a good accident investigation model with which to investigate software accidents.

Accident models from other fields are examined to determine their suitability for use in the field of software development. Although many existing models are very useful, all assume that the environment of the accident was operational. That is, there is a steady state of operations during which events or failures produce an accident. Software development and software operations are very different circumstances. Additionally, none of the existing models makes use of the considerable body of knowledge about software development. For these reasons a new system theoretic model is proposed, extending Leveson's model, STAMP. The model is briefly described and discussed, and areas or research arising from the model are briefly described.

Keywords: software development, software failure

ACM Classifications: D.2.0, K.5.2, K.4.2

INTRODUCTION

While there is no shortage of studies into the reasons why software projects fail (Ewusi-Mensah, 1997), the major risks of software development (Jones, 1994; Boehm and Basili, 2001; Wallace and Keil, 2004), or even the factors affecting project success (Cooke-Davies, 2002), the field of software engineering lacks a general model with which to investigate such failures. To date, studies have tended to be surveys of the factors thought to play some part in a failure. They serve as advice on what managers should consider when managing software development projects. While such advice is both welcome and valuable it is not intended to be used to investigate reasons for the failure of a specific.

Unlike some safety critical industries, software development itself has not been the subject of a judicial investigation and has not needed to develop models to guide such an investigation. Lacking such a model, a formal inquiry into the causes of a software failure is likely to adopt an investigation model used for other industries, possibly with some adaptation. Leveson (2002) proposed one such model, STAMP (Systems Theory Accident Modeling and Processes), primarily to investigate safety related accidents of any socio-technical system, including software systems. However, it is unclear how Leveson's model should be applied to software development other than in a general way.

Copyright© 2008, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that the JRPIT copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society Inc.

Manuscript received: 5 March 2007

Communicating Editor: Martin Purvis

ACCIDENT INVESTIGATION MODELS

Several researchers (Rasmussen, 1997; Reason, 1997; Turner and Pidgeon, 1997; Perrow, 1999; Leveson, 2002; Dekker, 2005) have argued that a simple model of accidents is insufficient for dealing with modern technology. A causal-chain model of accidents is useful to investigate the failure of a specific component through wear and tear, or the attribution of the cause can be established through application of a “but for” test. Given the cause, similar accidents can be prevented by checking the same component for wear and tear or other flaws such as structural cracks. However, it is a less useful model when investigating accidents whose causes are ultimately not due to physical weaknesses but are due to interactions between components or the failure of the system itself.

Driven by the need to find ways to prevent future accidents, the alternative models reject the simple causal chain model on several grounds. The first is that looking back along the causal chain requires a “stopping rule” to determine when to cease investigating deeper into the system which, it is argued, can be somewhat arbitrary in the choice of cause (Leveson, 2004). The second reason is that such investigative techniques tend to focus attention on the proximate event most closely associated with the accident and direct attention away from the latent, contributory causes (Turner and Pidgeon, 1997; Perrow, 1999).

Where, in the past, it may have been sufficient to seek direct causes of an accident modern socio-technical systems can produce accidents that are the result of the interaction of different parts of the system rather than a failure of any one part of the system (Rasmussen, 1997; Reason, 1997; Turner and Pidgeon, 1997; Perrow, 1999; Leveson, 2002; Dekker, 2005). Turner reviewed official investigations into non-natural disasters to arrive at a view that many disasters were man-made and entirely foreseeable. In a major contrast to causal models of accidents, Turner argued that the conditions for the disasters he investigated largely originated from decisions made by upper management.

In general terms, slips and errors which occur in the lower reaches of an organization are likely to be more modest in their consequences, for the higher the level at which an error originates, the greater chance it has of being compounded with and extended by other errors which it encounters in the course of its transmission down the hierarchy. (Turner and Pidgeon, 1997 p149)

So strongly did Turner believe that most, if not all, disasters could have been prevented if only those in positions to do so had paid attention to the many warning signs or had simply thought about the changes they were about to introduce that the original title of his book was “A Failure of Foresight”. This was later changed to “Man-Made Disasters”.

The view that there was ample evidence of impending disaster available if only someone paid it any attention appears to be shared by investigators other than Turner. However such hindsight bias has been criticised by several researchers, most notably Dekker (2005). Hindsight bias ignores the reality that most operational decisions are made under ambiguous circumstances based on sparse and ambiguous evidence (Reason, 1997; Dekker, 2005). Instead, Dekker argues, investigators must try hard to understand the circumstances of the time and put aside knowledge of the outcome.

To reason more fully about the interaction of different parts of a socio-technical system, several researchers (Rasmussen, 1997; Reason, 1997; Turner and Pidgeon, 1997; Leveson, 2002) have proposed a system theoretic model in which the system is expressed as a hierarchy of control levels. Each level of the hierarchy is considered to act on the level below it through the imposition of constraints and directions to achieve emergent properties and to receive feedback. While Turner

didn't formally propose a systems theory based hierarchical model, his view that different levels of a hierarchy contribute to accidents are consistent with the systems theory views. After several decades from multi-disciplinary research on industrial risk management Rasmussen (1997) observed that their efforts had moved in a full circle through several research disciplines and paradigms. Yet they found that the models created for the design of work systems were not very useful for the design of the total risk management system. A more useful model for considering total risk was a "top down, systems oriented approach based on system control theoretic concepts". This approach gave a control structure embedded in an adaptive socio-technical system. Such a model, shown in Figure 1, shows how different parties contribute to safety regardless of their organizational affiliations.

Each level of Rasmussen's model uses its judgement to impose constraints on the operations of the level below which provides feedback that, in turn, influences the judgement.

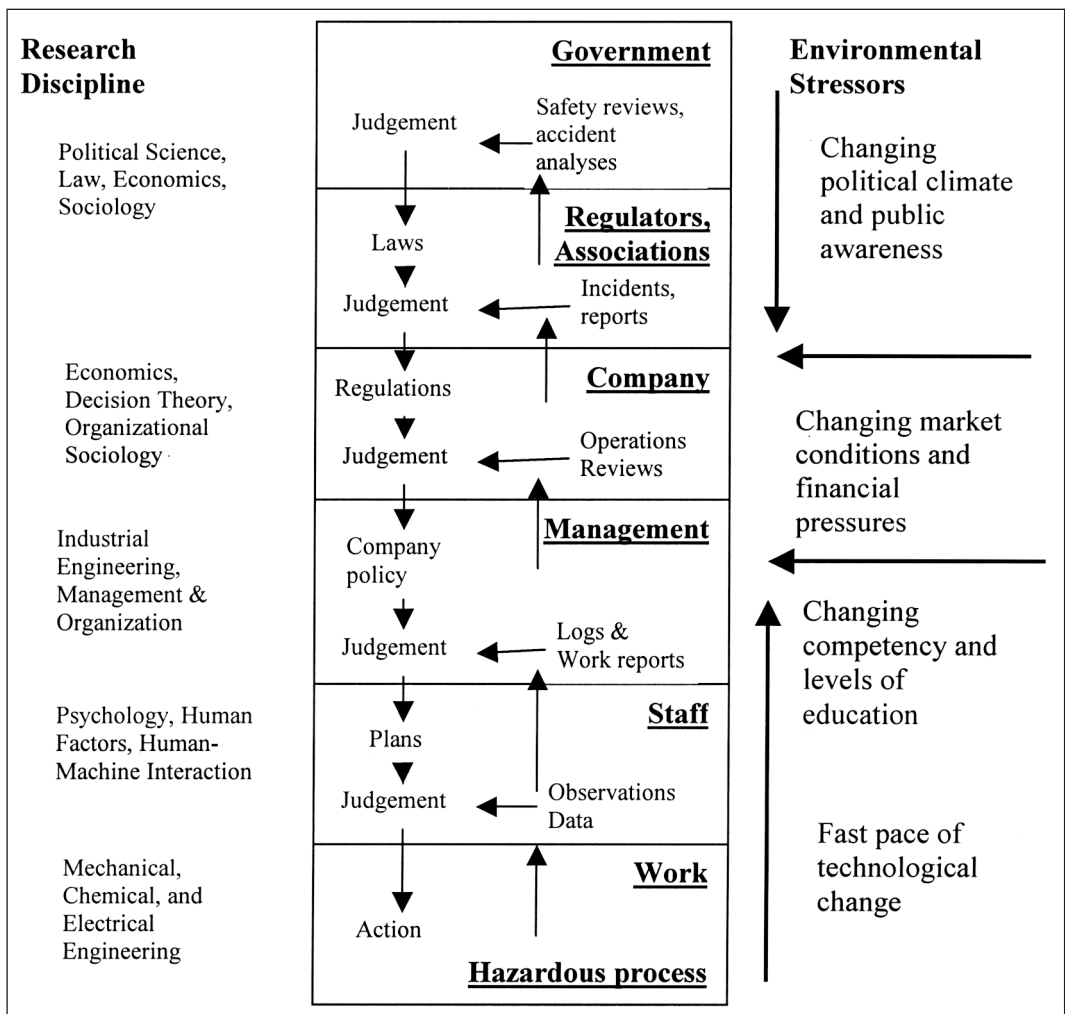


Figure 1: The socio-technical system involved in risk management (Rasmussen, 1997)

Leveson (2002) builds on the ideas used in the upper levels of Rasmussen's model but continues the control-theoretic approach down through the levels of a socio-technical system. In Leveson's model safety is achieved through the imposition of constraints and an accident is regarded as a failure of a constraint. Removal of the work processes assumes that the work processes themselves do not impose any constraints.

PURPOSES OF ACCIDENT INVESTIGATION MODELS

Retrospective investigations

The most obvious use for an accident investigation model is to investigate an accident that has already occurred. There, the model can guide the investigation toward matters to be investigated. Since it is already known that an accident has occurred, an investigation can be directed toward preventing similar accidents in future or allocating blame. Questions of compensation usually require that blame be apportioned. However, an investigation to apportion blame is fundamentally different to an investigation seeking to prevent similar accidents.

Questions of blame assume that a preventable error has occurred and that the exact sequence of events leading to the accident can be established (Leveson, 2002). Both Leveson (2002) and Perrow (1999) point out that many accidents involving modern complex systems are a consequence of interactions among components of the system and not attributable to a single cause nor the result of a single chain of events. Systems that involve software as one of the components are more likely to be complex than simple, and more likely to fail in complex ways than simple causal ways.

Hopkins (2000) proposes an interesting variant that acknowledges the differing motives for investigations. His proposal is to complete the investigation first, then to select appropriate causes rather than use the desired cause to direct the investigation.

Predictive investigations

Investigating a system as a preventative measure can be difficult because they require that the investigators consider how the system might fail. Perrow (1999) points out that accidents involving complex systems happen in ways that we did not or cannot imagine which suggests that predictive investigations may have difficulty detecting a possible, but unimaginable, failure. An exception might be a review conducted after an accident investigation that revealed previously unimagined circumstances. The unimaginable becomes possible, and investigators have a different perspective from which to review a system's vulnerabilities. Turner (1997) is not so circumspect when he pointed out that many designers assume benign usage when, in fact, users may not regard the system so benignly and simply smash controls, lock safety doors to prevent illegal entry, jam fire escapes with excess furniture and generally subvert the system.

LIMITATIONS OF EXISTING ACCIDENT INVESTIGATION MODELS

There are several attempts, other than Leveson's, to move beyond simple causal accident models (Isaac *et al*, 2002; Dien *et al*, 2004; Sklet, 2004). Most of these models are specifically concerned with traffic accidents or air transport rather than trying to be a general model of safety or accidents. Leveson's model does claim a wider applicability to safety in any system and specifically to safety in the software component of the system (Leveson, 2002).

However, there remain several concerns about the applicability of existing accident models to software accidents.

- The models concentrate on operations and do not explicitly deal with development of the software.
- They do not provide a framework within which the existing body of knowledge about software engineering can be incorporated.
- They require that decisions or control actions be oriented toward the single attribute of safety. This ignores all control actions that were not oriented toward safety e.g., a control action to correct project slippage.
- Investigations with the benefit of hindsight are limited in their ability to determine how the software or system came to be in the state in which the accident occurred.
- Existing accident investigation models concern accidents during operation of a control system and not accidents during development of the control system itself.

Although Leveson's model can be used to investigate software accidents, i.e., accidents that can largely be attributed to faults in the software, the documented investigations to date concern how the system was operated more than how the system was developed.

Although it is reasonable to apply knowledge of accident investigation in one context, that of accidents during operation, to another context, that of accidents during development, they are not the same context. When a system is in operation, even one as diverse as air transport, there are definite constraints on the situation, a sense of the limits within which the system must be controlled. Operating a power plant, driving a train, controlling road traffic, operating a factory production line are all operations of control systems. The objective is to keep the system and the situation under control.

Software development needs to be kept under control only in the sense that it must produce a working system. Keeping it under control will not produce software. Software development involves intellectual discovery and invention as much as it involves directing that discovery and invention toward producing a working system. Investigating only failures of control assumes that control is the most important activity, if not the only required activity. If existing models of accident investigation are to be extended to deal with failures of software development then the scope of investigation must be extended beyond that of system control.

REQUIREMENTS FOR A MODEL TO INVESTIGATE SOFTWARE DEVELOPMENT FAILURES

The primary objective is that the model be effective and efficient for investigating software accidents. This requirement is intended to prevent other potential purposes, such as modelling risk in software development, from diverting the model or compromising its ability to achieve its primary objective. It also helps decide what to include or exclude.

The model should also consider different types of cause. A finding of simple cause and direct effect is just as valid as a finding attributable to the more complex interaction between different parts of the socio-technical system or the environmental aspects that predisposed the organization to act in particular ways.

Studies of highly reliable organizations and teams have revealed that culture, organizational structure, redundancy and mindfulness all play a part on achieving that reliability (Rochlin *et al*, 1987; Weick, 1987; Bierly and Spender, 1995; Bigley and Roberts, 2001). The model must consider ways in which the socio-technical system is resistant to accidents as much as it investigates its vulnerabilities, and thus must incorporate elements of organizational structure, organizational and professional culture, and the different types of redundancy found to aid reliability.

The model should direct investigations toward the most likely causes quickly. To this end, the

model will likely be organized in layers so that investigation is not constrained to spend long periods delving into the detail of something unlikely to produce a result. It is acknowledged that such an approach risks overlooking a possible cause or directing attention toward the wrong cause.

The model must be able to deal with diverse project structures such as distributed or outsourced development as well as component based development.

INVESTIGATIVE APPROACHES

Software development is a complex and sometimes large endeavour where the proximate and contributory causes of a product or project failure could take an inordinate amount of time to locate and test. An appropriate investigative approach could introduce some much needed efficiency.

The many surveys of past software development project failures (Ewusi-Mensah, 1997; Boehm and Basili, 2001; Johnson *et al*, 2001) have produced lists of the main causes of software project failure. These could be taken as a starting point to examine if each of the factors was present during the project and, if so, to what degree it was present and how much it contributed to the failure. The difficulty with this approach is that investigating each factor is potentially time consuming and unrewarding. There is no guarantee that any one of the factors is present or, if it was, that it contributed to the failure. Additionally, the readily available lists of main causes are usually those at the top of a longer list with no assurance that the proximate cause could be found within the shorter or longer list.

Leveson (2002) orients her investigation model around safety by directing attention to safety related constraints. This allows the investigation to search for missing, inadequate or violated constraints at the different levels of the socio-technical system. While this is an efficient investigative approach, applying it to software development has some difficulties. The most obvious is that not all software is safety related so the number of constraints that must be considered grows and would quickly become unwieldy. Additionally, such an approach would need to consider the constraints related to the project and its conduct.

Turner proposed that an accident was the consequence of a failure of intent (Turner and Pidgeon, 1997 p151). Intent is something that can be interpreted at all levels of a socio-technical system. Specifically an investigation could consider what were the intentions, what were risks that the intentions would not be realised and how were those risks managed. Such an approach would take advantage of existing bodies of knowledge concerning risk management in software development (Charette, 1989; Jones, 1994; Roy, 2004; Wallace *et al*, 2004) as well as more general, organizational risk management (Rasmussen, 1997).

An intent based investigative approach also avoids needing to pay special attention to different organizational structures such as distributed or outsourced development because such differences are dealt with when the different levels of the organization establish and implement their intent, and manage their risks. Distributed development becomes an implementation of intent or risk management and examined as such, rather than be considered from the outset as something separate or different.

Focusing on intent and risks associated with that intent readily defines what needs to be considered and what can safely be ignored.

SYSTEMS MODEL OF SOFTWARE DEVELOPMENT FAILURE

Leveson's system theory based model of accidents was developed in response to the limitations of causal models of accident investigation, and to provide a means to investigate complex accidents that may have arisen from interaction among parts of the system, rather than from a simple failure of one of the system's components. It drew on the work of Rasmussen.

Although there may be some circumstances where an accident investigation should concern itself with levels beyond the organization, such as may be necessary when considering the contractual environment, at this stage it should be sufficient to model only those levels within the boundary of the organization, i.e., the strategic, tactical and operational levels, indicated in Figure 2 by a dashed box. Experience may prove this to be an unnecessary or undesirable limitation but it will be adopted at this stage.

Taking Leveson's model as a departure point, a model of software development can be proposed (Figure 2). The model is not proposed as a finished, fully tested model but as a starting point. The

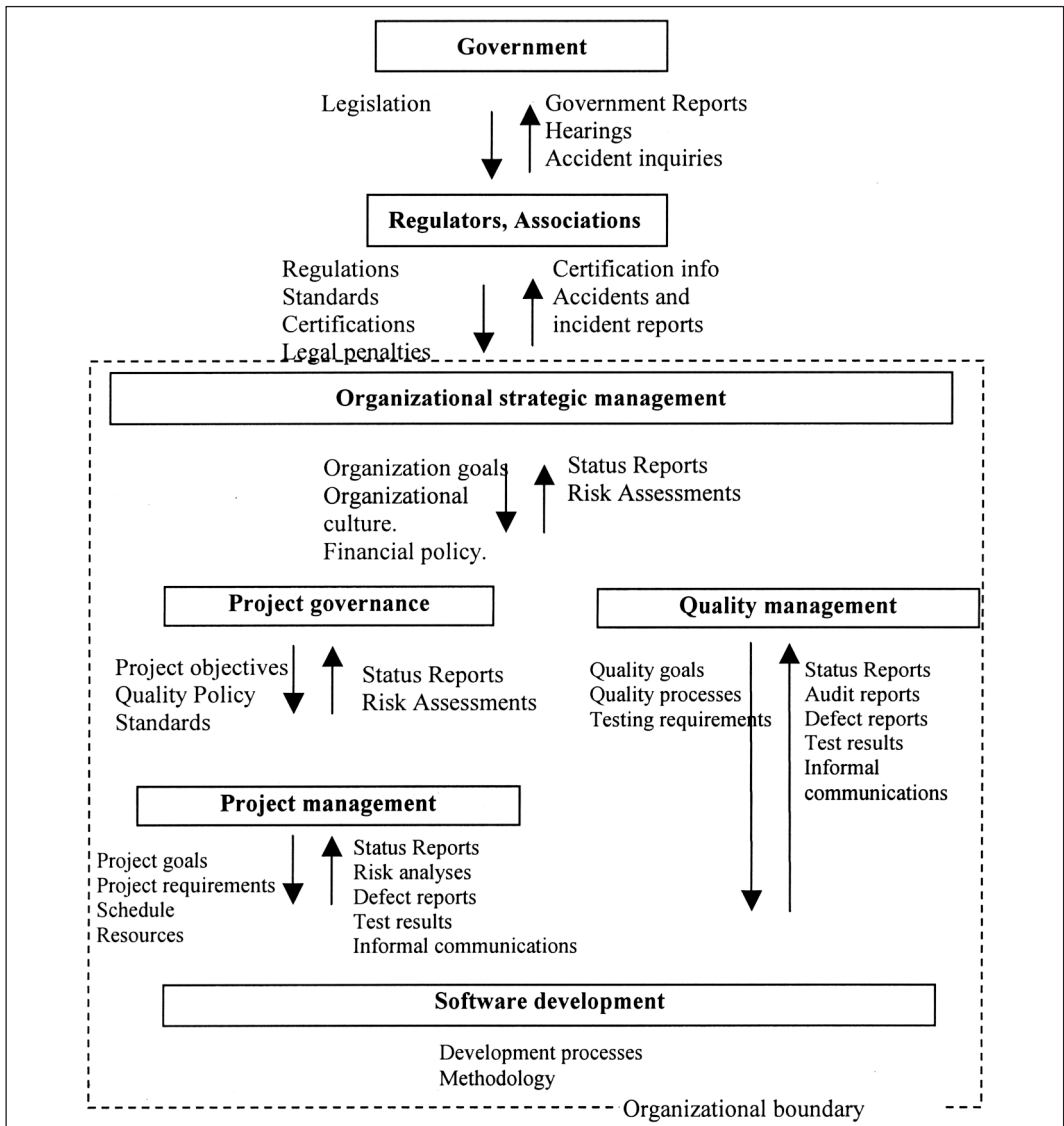


Figure 2: Systems theory model of software development failure – based on STAMP (Leveson, 2002)

model can be used to investigate software development accidents as well as point out areas where further research is needed. The model should remain a “living” model, to be updated in light of research outcomes or investigation experience.

An explanation of some terminology is in order. In this context the term “software development” can mean only those activities undertaken by the development team to develop the software or can also mean all of the activities undertaken by the organization related to software development. The industry does not consistently use specific terms for either meaning so the meaning of the term “software development” must be made clear when it is used. Similarly, the terms “software development”, “systems development” and “software systems development” can be used to mean the same thing or can be used to make subtle distinctions between different types of software related products. This research is concerned with software and is not concerned whether that software is part of a larger system that involves hardware and organizational processes. In this research the term “software” is intended to direct attention to the software part of any larger system or development.

The different elements of the proposed model will now be discussed.

GOVERNMENT

Governments pass laws, establish and fund regulatory bodies. In particular, the laws concerning fair trading, international trade, employment, taxation, occupational health and safety¹ and possibly more create the environment in which an organisation operates. Some of these legal or regulatory schemes may affect directly a software product or project’s requirements while others form part of the environment. The laws governing international trade are an example of the latter, unless the product happens to concern import tariffs or customs regulations. For the most part, the level of the government in the systems model is not expected to feature significantly in investigations.

REGULATORS, ASSOCIATIONS

The next level contains those who apply or interpret government regulations. This includes such organisations as the courts, industry associations, standards making bodies, professional associations and auditing organizations. Some national and international standards apply to software development. The best known of these are ISO 9001 (ISO 9001, 2000) and its guidance for software ISO 90003 (ISO 90003, 2004). The Integrated Capability Maturity Model (SEI, 2000), although not an international standard, is used as if it was one. Standards concerning specific aspects of software systems such as safety (ISO 61508, 1998) have arisen from the engineering field where there are already numerous safety-related standards, but few that directly apply to software.

Professional associations such as the Australian Computer Society play a role in defining the expected conduct of its members through, among other things, the Society’s code of ethics as well as the requirements for membership. Societies such as the Australian Institute of Engineers² can claim a stronger professional code through their “chartered” membership category which requires a verifiable demonstration of competence on top of the normal conditions of membership.

This layer of the model, as well as interpreting laws and promulgating regulations in support of those laws, provides an environment in which organizations, and people within those organizations, operate.

¹ These examples are relevant in Australia and are not meant to imply that all countries have the same laws and regulatory schemes. However, most countries have similar concerns but may be addressed in different ways.

² (See <http://www.ieaust.org.au/membership/professional.html>)

ORGANIZATIONAL STRATEGIC MANAGEMENT

The importance of strategic goals to direct IT projects has been pointed out in more recent texts, particularly those concerned with governance (Thorp, 1998; Weill and Ross, 2004). This level of an organization is not only concerned with strategy and strategic goals, even those concerned with IT projects, but there is emerging awareness that projects need to be guided by clear goals in addition to the normal project requirements.

PROJECT GOVERNANCE

Recently, there has been growing attention paid to the role of governance in and of organizations (e.g. Thorp, 1998; Weill and Ross, 2004). This has flowed down to governance of ICT projects (e.g. AS/NZS 8015, 2005). At this level of the model, concern focuses on governance of the project, or program of projects, and not on governance of ongoing ICT operations.

QUALITY MANAGEMENT

Quality management and management systems have been an established part of software development since the early 1990s. Such systems operate independently of project and although there are many different quality management systems, the majority of them are intended to satisfy ISO 9001 (ISO 9001, 2000). Although quality management systems may report to or inform those responsible for project governance, they are not usually directed by them.

PROJECT MANAGEMENT

Project management (e.g. McConnell, 1998; Project Management Institute, 2000; Cleland and Ireland, 2002; OGC, 2002) has a significant role in software development projects. The activities of project managers and the way they structure, plan and manage the project has a significant bearing on the project outcomes.

SOFTWARE DEVELOPMENT

At this level of the model concern is not with the day to day software development activities but with such matters as the choice and appropriateness of tools and methods used in the development, the design decisions made during the project and the management of technical matters during the project. For example, the correct functioning of interfaces between components is crucial. Nor is the model concerned with the particular method of software development. Some of the agile methods (Beck, 2000; Highsmith and Cockburn, 2001) may be appropriate in some circumstances whereas other circumstances may require plan-based methods.

CULTURE

The role of culture is not currently shown in the model. However, culture is the environment that affects how software development projects are carried out. The effects of culture on software development projects is not yet well researched but is largely acknowledged (Bierly and Spender, 1995; Vaughan, 1997; Chevrier, 2003). Culture that may affect software development projects in some way could be national culture (Hofstede, 1983), organizational culture (Martin, 2002) or professional culture (Gregory, 1983) or the interaction between any combination of these. Culture may affect the type of risks that are considered and how those risks are managed. Culture can also affect decision-making (Klein, 1998) although this has yet to be investigated within software development.

USING THE MODEL

As was foreshadowed in the previous section, this research proposes that intention and risk management at each level of the model be used to guide accident investigations. For most investigations it is unlikely to be appropriate or practical to investigate levels above that of organizational strategic management.

Starting with the organization's strategic objectives, at each level an investigation can consider

- What were, or should have been, the intentions of this level?
- How were those intentions implemented?
- What were the risks that the intentions would not be achieved?
- How those risks were managed?

Investigations can take advantage of any existing processes for risk management and governance that operate in the organization, and can draw on industry standards for governance, project management and software engineering as well as professional bodies of knowledge.

INTENT

Intention at any level of a socio-technical system may or may not be directly observable. At some levels there may be documented policies or strategies while at other levels there may be an absence of such things. Systems theory does not have a place for intent, everything must be interpreted in terms of constraints and emergent properties, directives and feedback. At each level of the system intent would be implemented with a combination of constraints and directives. Where there is a documented intent, the constraints and directives may be examined to determine whether or not they implemented the intent. When the intent is absent or tacit, it may be possible to infer the intent from the implemented constraints and directives.

Each level of the model, illustrated in Figure 2, has particular mechanisms to deal with intent. That is, they have methods, processes or procedures for eliciting, understanding, implementing and monitoring intent. What is vague at this stage are the details of how intent is expressed at each level and propagated to the level below. It would be unreasonable to expect that intent would be treated in the same way at each level because each level is likely to have different perspectives about intent and different ways of dealing with it. For example, project management is concerned with the overall goals of the project and would achieve the project intention in terms of project management whereas the software development level is interested in the software requirements and how those are to be implemented. The concept of project governance is relatively new and has yet to achieve broad consensus about how the organizational intent is to be achieved through such governance.

At each level, intent would be interpreted in terms of that level then implemented in some combination of previously established processes, technology, skills etc., and new processes, technology, skills etc.

It is a suggested task for future research to establish just how organizations implement organizational intent.

RISK MANAGEMENT

There is a considerable body of knowledge on risk management in software development (Charette, 1989; Jones, 1994; Karolak, 1996), risk in project management (Thomsett, 2002; Wallace and Keil, 2004), as well as the more general risks management for any organized endeavour (Clarke, 1989; Rasmussen, 1997; Harms-Ringdahl, 2004). The general principles of risk management have been expressed in national standards as well as international standards. There is also an existing body of

literature on general organizational risk management. The task of future research is to establish intentions and the risks associated with those intentions. This is to prevent the investigation examining any and all risks that may or may not be relevant. For example, all organizations are subject to the risk of occurrence of natural disasters, but such risks do not arise from the organizational intentions being realised through the project and should be excluded from the investigation.

FUTURE RESEARCH

Aside from the obvious research area of software accidents and their causes, the proposed model retains the view that software development is performed within a complex socio-technical system. Such a model is better suited to investigating complex problems than a simple causal model.

Possibly as important as the model's utility for investigation is the number of potential research areas that it reveals. The role of culture has been mentioned previously. However, there are also:

- Effect of culture on risk management.
- Effect of culture on decision making.
- Sense-making in software development.
- What errors arise from failures of governance?
- How do organizational structures affect software development projects?
- How does financial management, contractual management affect software development projects?
- Are software development methodologies vulnerable to some classes of error?
- What 'defenses-in-depth' are embodied in software development methodologies?

CONCLUSION

Accidents in software development projects and products are nothing new and neither are they likely to cease. As both software projects and products become larger and more complex, the effects of accidents are, as pointed out by Perrow (1999), likely to become more catastrophic. The current models employed to investigate software accidents do not appear to be suited to the task and, compared to some of the more advanced models in other fields, appear to be simple cause and effect models derived from current knowledge of program debugging, with the noted exception of Leveson's (2002) model. However, existing accident models from fields such as air transport (Dekker, 2002), general transport (Stoop, 2002; Sklet, 2004) or the previously mentioned STAMP model are suited to operational accidents rather than accidents that arise from development. Also, none of the current accident investigation models make any use of the existing body of knowledge of software development.

A new model is proposed for investigating accidents that arise from software development. Such accidents could manifest themselves in a failed project (financial loss) or failed product (loss of life or severe injury). The proposed model builds on previous accident models that are, in turn, based on the concepts of system theory. It is claimed that the proposed model will be well suited to investigate accidents in the complex socio-technical system within which software is developed.

In addition to its utility for investigating accidents, the proposed model raises several potential fields of fruitful research.

REFERENCES

- AS/NZS 8015 (2005): Corporate governance of information and communication technology
BECK, K. (2000): Extreme Programming Explained, Addison-Wesley, Boston
BIERLY, I., PAUL E. and SPENDER, J.-C. (1995): Culture and high reliability organizations: The case of the nuclear submarine, *Journal of Management*, 21(4):639-656.

A Model for Investigating Software Accidents

- BIGLEY, G.A. and ROBERTS, K.H. (2001): The incident command system: High-reliability organizing for complex and volatile task environments, *Academy of Management Journal*, 44(6):1281.
- BOEHM, B. and BASILI, V.R. (2001): Top 10 list [software development], *Computer*, 34(1):135-137.
- CHARETTE, R.N. (1989): Software engineering risk analysis and management, McGraw-Hill, New York.
- CHEVRIER, S. (2003): Cross-cultural management in multinational project groups, *Journal of World Business*, 38(2): 141-149.
- CLARKE, L.B. (1989): Acceptable risk?: Making decisions in a toxic environment, University of California Press, Berkeley and Los Angeles.
- CLELAND, D.I. and IRELAND, L.R. (2002): Project management: Strategic design and implementation, McGraw-Hill, 4th ed.
- COOKE-DAVIES, T. (2002): The "real" success factors on projects, *International Journal of Project Management*, 20(3):185-190.
- DEKKER, S. (2002): The field guide to human error investigations, Ashgate Publishing Limited, Aldershot
- DEKKER, S.W.A. (2005): Ten questions about human error: A new view of human factors and system safety, Lawrence Erlbaum, Mahwah, NJ.
- DIEN, Y., LLORY, M. and MONTMAYEUL, R. (2004): Organisational accidents investigation methodology and lessons learned, *Journal of Hazardous Materials*, 111(1-3):147-153.
- EWUSI-MENSAH, K. (1997): Critical issues in abandoned information systems development projects, *Communications of the ACM*, 40(9):74-80.
- GREGORY, K.L. (1983): Native-view paradigms: Multiple cultures and culture conflicts in organisations. *Administrative Science Quarterly*, 28(3):359-376.
- HARMS-RINGDAHL, L. (2004): Relationships between accident investigations, risk analysis, and safety management, *Journal of Hazardous Materials*, 111(1-3):13-19.
- HIGHSMITH, J. and COCKBURN, A. (2001): Agile software development: the business of innovation, *Computer*, 34(9):120-127.
- HOFSTEDE, G. (1983): National Cultures in Four Dimensions: A research based theory of cultural differences among nations, *International Studies of Management and Organization*, 13(1-2):46-74.
- HOPKINS, A. (2000): Lessons from Longford: The Esso gas plant explosion, CCH Australia Limited, Sydney
- ISAAC, A., SHORROCK, S.T. and KIRWAN, B. (2002): Human error in European air traffic management: the HERA project, *Reliability Engineering & System Safety*, 75(2):257-272.
- ISO 9001 (2000): Quality management systems - Requirements
- ISO 61508 (1998): Functional safety of electrical/electronic/programmable electronic safety-related systems
- ISO 90003 (2004): Software engineering - Guidelines for the application of ISO 9001:2000 to computer software
- JOHNSON, J., BOUCHER, K.D., CONNERS, K. and ROBINSON, J. (2001): *Collaborating on Project Success*, Available: www.softwaremag.com/archive/2001feb/CollaborativeMgt.html, accessed July 17, 2002
- JONES, C. (1994): Assessment and control of software risks, Prentice-Hall Inc
- KAROLAK, D.W. (1996): Software engineering risk management, IEEE Computer Society, Los Alamitos, CA
- KLEIN, G. (1998): Sources of power: How people make decisions, MIT Press, Boston.
- LEVESON, N. (2002): A new accident model for engineering safer systems, *MIT Engineering Systems Division Internal Symposium*, Boston: 1-27, MIT
- LEVESON, N. (2004): A systems theoretic approach to safety in software-intensive systems, *IEEE Transactions on Dependable and secure systems*, 1(1):66-86.
- MARTIN, J. (2002): Organizational culture: Mapping the terrain, Sage Publications, Thousand Oaks
- McCONNELL, S. (1998): Software project survival guide, Microsoft Press
- OGC (2002): Managing successful projects with PRINCE2., Office of Government Commerce, London, 3rd ed
- PERROW, C. (1999): Normal accidents: Living with high-risk technologies, Princeton University Press, Princeton originally published Basic Books, 1984
- PROJECT MANAGEMENT INSTITUTE (2000): *A Guide to the Project Management Body of Knowledge*, Project Management Institute
- RASMUSSEN, J. (1997): Risk management in a dynamic society: a modelling problem, *Safety Science*, 27(2-3):183-213.
- REASON, J. (1997): Managing the risks of organizational accidents, Ashgate Publishing Company, Brookfield
- ROCHLIN, G.I., PORTE, T.R.L. and ROBERTS, K.H. (1987): The self-designing high-reliability organization: Aircraft carrier flight operations at sea, *Naval War College Review*, 51(3):97.
- ROY, G. G. (2004): A risk management framework for software engineering practice, *Australian Software Engineering Conference*: 60-67.
- SEI (2000): *CMMI for Systems Engineering/Software Engineering, Version 1.02*, Carnegie Mellon University/Software Engineering Institute, Pittsburgh, CMU/SEI-2000-TR-019
- SKLET, S. (2004): Comparison of some selected methods for accident investigation, *Journal of Hazardous Materials*, 111(1-3):29-37.
- STOOP, J. A. (2002): Accident investigations: trends, paradoxes and opportunities, *International Journal of Emergency Management*, 1(2):170-182.

- THOMSETT, R. (2002): *Risk in Projects: The Total Tool Set*, Available: http://www.thomsett.com.au/main/articles/risk_0404/risk0404_toc.htm, accessed 17 August 2004
- THORP, J. (1998): *The information paradox*, McGraw-Hill, Toronto
- TURNER, B.A. and PIDGEON, N.F. (1997): *Man-made disasters*, Wykeham Publications, Oxford, 2nd ed originally published 1978
- VAUGHAN, D. (1997): *The Challenger launch decision : Risky technology, culture, and deviance at NASA*, University of Chicago Press, Chicago
- WALLACE, L. and KEIL, M. (2004): Software project risks and their effect on outcomes, *Communications of the ACM*, 47(4):69-73.
- WALLACE, L., KEIL, M. and RAI, A. (2004): Understanding software project risk: a cluster analysis, *Information & Management*, 42(1):115-125.
- WEICK, K.E. (1987): Organizational Culture as a Source of High Reliability. *California Management Review*, 29(2):112-127.
- WEILL, P. and ROSS, J.W. (2004): *IT governance: How top performers manage IT decision rights for superior results*, Harvard Business School Press, Boston

BIOGRAPHICAL NOTES

Tom McBride is a senior lecturer at the Faculty of Information Technology at the University of Technology, Sydney. He has been involved in developing and teaching a number of subjects, including advanced data management, software engineering, software architecture and systems development. His research interests include software engineering and the investigation of software accidents, software process assessment, service management process assessment and decision-making in software development. He has worked in the software industry for more than 25 years with most recent experience in both project management and quality assurance. He is involved with software engineering standards development with both Standards Australia and the International Standards Organization.



Tom McBride