Toward An Integrated Framework of Software Project Threats

Paul L Bannerman
NICTA, Australian Technology Park, Sydney, Australia
paul.bannerman@nicta.com.au

ABSTRACT
This paper proposes an integrated threat management framework to improve outcomes of software projects. Current best practice prescribes risk management and issue management to control threats. However, these processes cover only part of the spectrum of uncertainty. A broader framework of threat management is proposed that integrates issue management, risk management and crisis management. Case examples and steps to transition beyond risk management are also provided. Implications for research and practice are discussed. The framework provides a basis to extend efforts in research and practice to improve the capability of organizations to manage uncertainty and improve project results.

Keywords: Threat management, risk management, issue management, crisis management, software projects.

1. INTRODUCTION
Herbert Simon argues that individual and organizational behavior is boundedly rational, constrained to satisficing (seeking a ‘good enough’ result) rather than maximizing (seeking the ‘best’ result) because of limited knowledge of the environment and limited capacity to choose a response [52]. Software projects are especially subject to bounded rationality, induced by cost and schedule constraints, resource limitations, and organizational and technological uncertainty. The question arises: Are current project risk management practices ‘good enough’ or is there still room for improvement?

While there have been many research advances over the last forty years, software projects continue to exhibit variable performance in practice, featuring notable successes and failures [14]. The monetary cost of poor performance and failure is high, and the value of missed benefits is substantial.

Risk management is proposed as a technique that can improve software project outcomes [11]. Many approaches can be found in practice and the literature. Evidence suggests, however, that risk management research lags the needs of practice, and risk management adoption in practice lags the prescriptions of research, pointing to opportunities for improvement [7, 8]. For example, it was found that risk management cuters only for known or foreseeable threats; it tightly couples the risk event with the risk consequence, ignoring the mediating influence of organization-specific vulnerabilities and capabilities to respond; managers prefer qualitative assessments of risk over quantitative, and; risk identification is highly subjective, biased towards factors outside of the assessor’s domain of responsibility. This suggests that current risk management may not yet be ‘good enough’, with both opportunity and need to improve the contribution of research and practice to project outcomes.

Consider the following Australian cases:

- In 2002, Sydney Water’s Customer Information and Billing System project was terminated at a loss of $61 million. An Auditor-General’s review of the project found, among other problems, that project planning and specifications were inadequate, resulting in many change requests; project governance was deficient; risk management was not effective; there was a cultural belief that outsourcing projects transferred risk to the contractor; important contingencies were not budgeted; and an independent ‘health check’ review that made significant recommendations to improve processes was ignored [34].
- In 2004, a joint Department of Family & Community Services and Centrelink project was terminated before completion after six years of development and expending over $64 million. The Edge project was building a system for processing claims and payments for family-related entitlements. An audit review found, among other problems, that a significant technical risk identified in the business case had inappropriate and ineffective mitigation strategies. Furthermore, this risk proved to be a significant factor in delaying the project [3].
- In 2005, implementation of Australian Customs’ Integrated Cargo System’s Import module caused major disruptions to the movement of cargo through Australian ports, significantly impacting Australia’s supply chain and international trading environment. This precipitated a crisis for which it was not prepared. The system was over two years late and project costs had increased from $30 million to $205 million. An audit review of the project found, among other problems, that project reports consistently rated risks as ‘extreme’ or ‘high’ but there was no evidence of these risks being monitored or mitigated; there was no implementation strategy incorporating a risk assessment of the implementation; and problem and incident management processes were insufficient for the high number of defects and change requests that were generated during the development [4].

Not all software project threats or ‘failures’ are the direct result of software and systems engineering practices, although many are. Others arise due to projects, the organizational vehicle through which new software is usually developed and delivered to the client or internal business user. Yet other threats arise due to the context of organizational change in which software projects operate (software projects are usually about delivering some form of technology-based change to an organization). Finally, others result from the external economic, technological, socio-political and regulatory environments that tend to impact organizations and their operations from time to time. Regardless of the source, these threats all have the potential to impact the outcome of system development activity if it is project-based.

It is often argued that many of these software project threats have such a low probability that spending time and money on them is
not justifiable, even if it were possible to prepare for the unknown. Also, unforeseen threats are often ‘acts of god’ and no rational organization expects its project team to be accountable for the outcomes of such events. This argument has some validity, especially for small projects and projects that have a limited focus on isolated functional requirements rather than enterprise-wide innovations. However, the above examples from the public sector and many classic cases in the literature suggest that there is a need to improve the management of threats to software projects. These classic project failure cases include, for example, Bank of America’s MasterNet trust management system; French Railway’s Socrates reservation system; London Stock Exchange’s Taurus equities settlement system; California DMV’s registration system; American Airline’s Confirm travel reservation system; Westpac’s CS90 banking system; Denver International Airport’s baggage handling system; London Ambulance Service’s computer-aided dispatch system. For more examples, see Charette’s ‘Software Hall of Shame’ [14].

This paper extends prior research [7, 8] by proposing a broader view of threats than is currently found in software project research as an approach to responding to the current limitations of risk management. It proposes a conceptual framework of software project threats that comprises multiple processes (issue management, risk management, crisis management and threat management) that span a continuum of threats from factors that are known and materialized to those that are totally unforeseen. The value of this theoretical framework and its associated processes is in refocusing research attention on the problems of threats and uncertainty in software projects and providing practitioners with improved ways to manage uncertainty and help deliver more successful project outcomes for stakeholders.

The key contributions of the paper are that, first, it portrays threat management in a broader context of vulnerability than that which is currently associated with risk management. Second, it provides an integrative view of the literature on threat management from diverse domains. Third, it proposes a conceptual framework that integrates complementary threat management processes across the uncertainty-uncertainty spectrum that software projects face.

The paper is structured as follows. In Section 2, contributing concepts and approaches from the literature are review. The integrated threat management framework is then proposed in Section 3. Application of the framework is illustrated with case examples in Section 4, and transitioning steps in Section 5, before concluding with a discussion of the contributions of the proposal.

2. LITERATURE BACKGROUND

This section briefly reviews contributing concepts to an integrated framework of threat management. These include risk management, issue management, crisis management, disaster management and emergency management.

2.1 Risk Management

The foundations of risk management in software engineering were laid by [11] and [13]. Typically, in software engineering, risk is defined as “the probability of incurring a loss or enduring a negative impact” [19]. For each identified risk factor, risk exposure is calculated as the product of the probability of its realization and the magnitude of loss of the associated unsatisfactory outcome. Risk management is defined functionally as “an organized process for identifying and handling risk factors; [this] includes initial identification and handling of risk factors as well as continuous risk management” [19]. Continuous risk management is “the process of analyzing the progress of a planned activity, project, or program on a periodic, ongoing basis and handling identified risk factors; [this] includes developing options and fallback positions to permit alternative solutions to reduce the impact if a risk factor becomes a problem” [19].

While risk reduction is an implicit objective in the design of many software engineering processes and practices (such as the spiral model of software development, software quality assurance, test-driven development and agile development), risk creation is inherent in others (such as the waterfall and rapid application development methods). Explicit risk management is mostly practiced in the context of software projects as a sub-process of project management. Software projects have long been recognized as high-risk ventures prone to failure [1, 12]. Risk management in software projects is important to help avoid disasters and rework, to focus and balance effort, and to stimulate win-win outcomes [11]. Normative approaches to practical risk management in software projects are described by [15], [18] and [22]. Many practice-based approaches also exist (e.g., in Prince2, CMMI, COBIT, ITIL, ITGI, NIST, COSO). Iterative risk management steps usually include risk identification, risk analysis, risk response, and risk monitoring and control [51, 6].

Other approaches to risk management in the research literature, for example, emphasize early development lifecycle risk avoidance in favor of late lifecycle testing to eliminate software defects [2]; scenario-based risk management [10]; modeling operational risks via Bayesian networks [20]; software risks within a socio-technical model of organizational change [29, 30]; life cycle-based enterprise security risk management [16]; real options approach to managing incomplete knowledge in projects [38]; and contingency modeling of software project risk management in which the outcome of a development project is hypothesized to be influenced by the fit between the project’s risk and how that risk is managed [9].

While the literature reports that risk management can significantly improve software project outcomes [14, 26, 43, 44, 56], other studies find that risk management is not always effectively applied in practice [23, 33, 39, 42, 43]. That is, it is more an espoused theory than a theory in use. It is also found that risk management as conceived in the literature lags the needs of practice to manage the uncertainties that confront and impact the performance of software projects [8]. A major limitation in particular is that risk management encompasses only known or foreseeable threats. It does not account for unforeseeable threats because a threat has to be foreseen in order to assign it a probability of occurrence and impact. On this basis, risk management is only a partial solution to managing software project threats.

It is unrealistic to expect that every potential threat to a software development project effort can be foreseen. Furthermore, research indicates that stakeholder groups perceive risks differently from each other, identifying as risks threats that arise outside of their domains of responsibility [27, 46]. This can significantly increase the likelihood that not even all relevant foreseeable threats are identified, especially if risk identification, monitoring and control
are left to a specific stakeholder group or individual, such as the project manager, which is often the case in practice.

To counter these deficiencies, it could be argued that if the risk monitoring and review life cycle is iterated continuously, as espoused by risk management best practice, then it is likely that previously unforeseen or ignored threats will be identified as they come into view during a project. However, dependence on coincidental threat recognition during risk monitoring is a high risk tactic in itself. Unforeseen threats are often preceded by warning signals, but not always. Also, the tactic assumes open scanning for new threats rather than just monitoring already identified and prioritized threats from the risk register. Frequently, risk monitoring is limited to the project manager updating the risk register prior to a steering committee meeting, rather than a formal risk review with a panel of representative project stakeholders. Finally, and most significantly, coincidental threat identification via risk management enables none of the preparation for a major disruption that an effective response capability requires.

This suggests that project threat management would benefit from an approach that augments risk management with other important processes that also have threat reduction and control as their objective. These include, for example, issue management and crisis management [8]. Indeed, [53] argues that risk management is a subset of a much wider crisis management process. In the organizational context, risk management is an attempt to deal with threats at the operational level of activities. Failure to manage small operational risks can aggregate upwards to generate higher level crises, failures and/or losses. However, as the project failure cases show, even at the operational project level, other threats can arise than those immediately catered for by risk management.

Following, we briefly review other threat management processes found in the literature that may also impact software engineering-based projects but which are not found in mainstream software project management research or methodologies.

### 2.2 Issue Management

In the corporate context, issue management has roots in public policy and social responsibility [57, 24] and strategic management [5]. In the former case, an issue is “a condition or event, either internal or external to the organization which, if it continues, will have a significant effect on the functioning or performance of the organization or on its future interests” (cited in [25]), and issue management is “the process by which the corporation can identify, evaluate, and respond to those social and political issues which may impact significantly upon it” ([57], p124). In this context, the overall purpose of issue management is to align organizational activities and stakeholder expectations by acting as an early warning system to minimize surprises associated with social and political changes, and enabling more systematic and effective responses to specific problems. It is an organizational response mechanism rather than a means of creating social change or controlling societal events.

In the strategic management context, strategic issues are “developments or trends that emerge from an organization’s internal or external environments [that] are perceived to have the potential to affect an organization’s performance” ([17], p355). Strategic issue management is a composite set of “organizational procedures, routines, personnel, and processes devoted to perceiving, analyzing, and responding to strategic issues” ([17], p355). Strategic issue management is often IT-enabled. The aim of strategic issue management is to improve the alignment of the organization with its environment and enhance the organization’s capacity to adapt and learn.

In software engineering and related information systems fields, issue management has not attracted an academic research focus other than to distinguish an issue from a risk as something that has already occurred. It is more a practice-level consideration and an application for developers of integrated software development environment products. In practice, issue management commonly arises in the form of defect tracking systems, problem reporting systems, change request systems, and help desk support systems. In these contexts, an issue often encompasses a range of concerns, some of which may need investigation to qualify them as a software defect, a non-defect problem, an action for the future, or a potential future problem (that is, a risk). Issues are usually managed with the help of an IT-based workflow system that provides functionality to submit, assign, evaluate, reassign (if necessary), resolve, and report issues as well as track their status and notify progress and resolution. More advanced systems will enable software defects to be tracked and managed independently of non-defect issues and risks, after they have been qualified, either within the same system or through integration with separate defect tracking, change request, risk management, and issue management systems.

It can be seen from this description that issue management in software engineering practice shares many characteristics with its counterpart in the social, public policy and strategic management research literature. An issue is a condition or event that needs to be acted upon and/or qualified now, before it has a significant effect on operational performance. The response is a controlled one, usually facilitated by a workflow system. Issues differ from risks in that the latter are potential future problems that may or may not require action now, other than monitoring, to mitigate their potential effects. Issues are realized problems that require action (either qualification or resolution) now. In the extreme case, an issue might be so urgent and threatening that it immediately qualifies as a crisis.

### 2.3 Crisis, Disaster, Emergency Management

Crisis, disaster and emergency management are considered together because, first, while they are discussed extensively in the research literature, this is rarely in the context of software engineering-related activities and, second, there is little consensus in the literature on the definition of these terms and their differences. Often they are used synonymously, especially in the case of crisis and disaster. Alternatively, crisis is favored as the process (crisis management) and disaster is favored as the triggering event. Accordingly, some researchers view a disaster as a subordinate condition that precedes a crisis while others take the reverse position (a crisis precedes and is subordinate to a disaster). Emergency management, on the other hand, tends to relate to the civil or governmental tactical response to crisis or disaster events, whether they are minor or major. In all cases, an overlap with elements of risk and issue management is evident.

One of the problems in reaching definitional consensus is that what constitutes a crisis, disaster or emergency varies with the
situational context. Another is that they are inter-disciplinary concerns, discussed in research fields as diverse as civil/national defense, public relations, and corporate/business continuity with little cross-disciplinary discourse.

It is in the corporate/business continuity context that these topics become relevant in the IT domain. At present, their main IT application is in Disaster Recovery Planning (DRP), a component of Business Continuity Planning (BCP). This comprises processes for recovering computer-based data and operations following a ‘disaster’ event of some kind. Responding to the ‘disaster’ may represent an ‘emergency’, calling into action contingency plans. Until recovery is complete, this may constitute a ‘crisis’ situation for the organization.

The aim of this review is not to be systematic or exhaustive but to distil characteristics of these fields of study that might be relevant and beneficial to software projects. Each management process is briefly considered in turn.

Crisis management. A crisis has been defined as a low probability high consequence event that threatens the most fundamental goals of an organization [60, 35]. It is characterized by ambiguity of cause, effects and means of resolution, and by a belief that decisions must be made swiftly. Other definitions can be found in [27]. Dimensions of crisis that are often cited in the literature include: high magnitude; requires immediate attention; contains an element of surprise; the need for taking action; may be outside the organization’s complete control; and poses a threat to the organization’s reputation and/or viability [37].

In organizations, crises can arise from internal or external causes, usually socio-technical events, and progress through a series of phases [21]: prodromal crisis (pre-incident warning phase of an impending crisis); acute crisis (the crisis incident itself); chronic crisis (post-incident phase of recovery and/or clean-up), and; crisis resolution (recovery of an operational status).

Crisis management is a systematic effort to avert crises or effectively manage those that do occur [35]. Crisis management is both a process and a mindset that drives an organization’s daily decisions and actions [36]. Indeed, it is argued that crisis management should be integrated with strategic management as complementary organizational processes [40].

Organizations tend to be crisis-prone or crisis-prepared [37]. Crisis-prone organizations make only limited or no preparation for crises. Crisis-prepared organizations invest in developing organizational capabilities that enable them to prepare for, respond to, and manage their way through any crisis that may arise. These capabilities are built on an investment in learning and development of organizational resources. Often, few organizations learn from crises because they are seen as unique or exceptional incidents [45]. In such cases, the priority is on recovery rather than exploring how the organization can leverage the experience for the future. Organizations are systems of production as well as systems of destruction [50]. They must be managed for improved productivity as well as designed, structured and managed to reduce their destructive potential.

In sum, what is common to these views of crisis management is the emergence of a critical threat, a sense of urgency to respond to avert a major organizational impact, and the development of capabilities to respond.

In software engineering, crisis and crisis management are not developed concepts, although [19] provides the following definitions: a crisis is “a critical state of affairs in which a decisive, probably undesirable outcome is impending”, and crisis management is “steps to take when a contingency plan doesn’t solve the associated problem”.

Disaster management. As indicated above, the word ‘crisis’ is often used interchangeably with ‘disaster’. Also, as with crisis, there is no generally accepted definition of disaster [47]. There tends to be greater clarity with ‘natural’ disasters [48, 49] but our interest here is in organizationally-based disasters. Four types of ‘organizationally-induced crises’ are identified [41]: socio-technical disasters, ecosystem disasters, business-economic failures, and sociopathic attacks. Our interest in this paper is restricted to the first type (ultimately, all four types may impact software projects, but, initially, we focus on events that may more likely arise from or directly impact software projects).

According to [54], the causal features common to organizational disasters are “rigidities in institutional beliefs, distracting decaying phenomena, neglect of outside complaints, multiple information-handling difficulties, exacerbation of hazards by strangers, failure to comply with regulations, and a tendency to minimize emergent danger. Such features form part of the incubation stage in a sequence of disaster development, accumulating unnoticed until a precipitating event leads to the onset of the disaster and a degree of cultural collapse” ([54], p378). Causality is more succinctly attributed to failures in two sets of interacting factors [50]: HOT (human, organizational, and technical) factors that lead to triggering events and compound the adverse effects of disasters when they occur; and RIP (regulatory, infrastructural, and preparedness) factors, which allow disasters to happen as well as reduce the effectiveness of disaster mitigation.

Technological disasters are divided into three stages [49], each with associated disaster management processes: pre-disaster stage (prevention and mitigation); disaster stage (response and relief), and; post disaster stage (reconstruction and rehabilitation).

Emergency management. Emergencies are extreme or exceptional events that are rare, uncertain and potentially have high or broad consequences [31]. Their rarity makes them difficult to model and predict, and the uncertainty surrounding their antecedents and consequences makes it difficult to respond to them with pre-planned contingencies. Much of the focus of emergency management is on planning and response to emergency events (often called crises or disasters) in practice. Broadly, emergencies encompass ‘routine’ day-to-day events, such as motor vehicle accidents, fire or ambulance calls; major natural disasters such as flooding or earthquake; major man-made disasters such as ‘pilot error’ accidents or a terrorist attack; and socio-technical disasters such as major systems failures or industrial disasters. Our primary interest remains in the last category (socio-technical disasters).

Emergency management includes mitigation (to prevent or lessen the impact of disaster), preparation (emergency planning, organizing and training), response activities (search and rescue), and recovery (post-event restoration of services and operations) [58].
The literature includes extensive IT-based research on emergency management information systems (e.g., [55]), support for emergency operations, simulation of emergency scenarios, and other technical contributions. See, for example, the March 2007 special issue of Communications of the ACM, which is devoted to emerging trends and technologies in emergency response information systems.

A common characteristic of emergency management in practice is that it often requires managers to develop and deploy new plans and operating procedures nearly simultaneously, in response to highly non-routine situations and under conditions of time, risk and resource constraint [31]. Consequently, the abilities of decision makers to analyze and improvise are highly relevant. In the event that the emergency is novel, an appropriate management plan needs to be developed and deployed in real-time. Over time, these response capabilities can become part of the organization’s knowledge base, increasing its ability to handle future events.

Accepting the high level of overlap between crises, disasters and emergencies in the literature, in this paper, the concept of crisis management will be taken to include disasters and emergencies, depending on the nature of the crisis.

In the next section, many of these concepts are carried forward in the development of an integrated threat management framework for software projects.

3. INTEGRATED THREAT FRAMEWORK

This section develops the proposed conceptual framework. First, the potential scope of software project threats is described using a matrix of the dimensions of what is known or not known about contingent events (that is, potential threats). Second, the main management processes in the framework are described and, third, response actions within each management process are outlined.

3.1 Threat Scope

We have argued that the fundamental components of the challenge of determining software project outcomes are: uncertainty that all contingent variables are known and under control; the existence of threats to achieving the desired outcome; and bounded rationality that limits knowledge of these contingent threats and responses to them. Two figures help conceptualize the scope of this challenge.

Conceptually, Figure 1 maps the dimensions of awareness and understanding of knowledge from what is known about potential project impacts (certainty) to what is unknown (uncertainty). The bottom two quadrants account for management processes that software project ‘best practice’ already prescribes. The known-unknowns quadrant (things we know we don’t know) is the traditional domain of risk management. In this category, potential threats are identified; risk exposures calculated (quantitatively or qualitatively); relative priorities assessed; mitigation actions and/or contingencies are planned and implemented; and statuses are monitored. The known-knowns quadrant (things we know we know) is the traditional domain of issue management. As noted earlier, issue management tends to be vested in defect tracking systems, problem reporting systems, change request systems, and help desk support systems in software engineering. Management processes typically include qualification, assessment, assignment and resolution of issues, as well as data recording, tracking and reporting via a supporting information system.

Organizations that ‘firefight’ their way through threats or issues as they arise rather than practice risk or issue management face uncertainties characterized by the top two quadrants.

The top two quadrants of Figure 1 account for things that are not foreseen, regardless of the magnitude (or potential magnitude) of their impact and whether they are known to others (unknown-knowns) or not (unknown-unknowns). They represent potential issues or crises/disasters/emergencies if they occur, depending upon the scale of impact. However, since they are not foreseen, often nothing is done in advance to prepare for their arrival. Certainly there is currently no prescription for crisis management behavior in software projects. When events of these kinds do occur, it is often with a high element of surprise. If no mechanisms are in place to respond to them, significant disruption can ensue or a major impact might be realized before a response can be determined and enacted.

<table>
<thead>
<tr>
<th>Unknown</th>
<th>Known</th>
</tr>
</thead>
<tbody>
<tr>
<td>Things we don't</td>
<td>Things we know</td>
</tr>
<tr>
<td>know we know</td>
<td>we know</td>
</tr>
</tbody>
</table>

**Figure 1. Dimensions of knowledge and uncertainty**

For the purposes of this paper, the distinction between the top two quadrants is ignored. Both may result in a threat that can immediately impact a software project and, in the case of a crisis or disaster event, might be extremely disruptive or even terminal for the project.

Figure 2 extends this view with an integrated framework of threat management along two dimensions, reflecting whether the threat has materialized or remains a possibility and whether the response to the threat is controlled or reactive.

Again, the bottom two quadrants reflect established best practice in software projects: a controlled response to a possible threat, accounted for by risk management, and a controlled response to a materialized threat via issue management.

In contrast, the top left quadrant represents a management process that is not yet mainstream in software projects: reactive responses to realized threats which necessitate crisis management,
especially if the associated impact or potential impact is significant (such as in the case of a socio-technical disaster).

This integrated framework of threat management is explicated further in Figure 3 where indicative core threat responses are listed for each of the subordinate management processes (that is, Risk, Issue, Crisis or Disaster, and Unknown). Actual response sub-processes used (the column titled Responses in the Figure) may vary in practice to match organization-specific circumstances. The responses listed are typical of those described in the literature. Each is outlined following.

**Risk responses (Known-Unknowns).** The focus here is on risk responses rather than the full cycle of familiar iterative risk management steps (which typically include identify, analyze, respond, and monitor risks). Alternative response strategies to software project risks are widely found in the literature [8]. These strategies typically aim to reduce or eliminate the likelihood of an identified threat occurring and/or limit the impact if it is realized. Options include avoid the threat by using software engineering practices and project designs that obviate the threat; transfer the risk to a third party via insurance, contracts, warranties and/or outsourcing; or accept the threat as a potential project impact. Acceptance may include one or more of several active or passive risk response options, namely: mitigate the risk through actions designed to reduce the threat (e.g., by reducing the likelihood and/or impact if the risk is realized); establish a contingency (such as an action plan or reserve funds) to use if the threat materializes; monitor the risk to determine whether the threat increases or disappears by itself; or do nothing because the threat is so low or specific that no cost-effective response can be made in advance of it occurring.

**Issue responses (Known-Knows).** It was found in Section 2 that there is no uniform approach to issue management in software engineering. However, system-based practices have a number of actions in common. The first is to qualify the issue. For example, is it a requirements issue, software defect, design problem, management problem or, indeed, only a potential problem (that is, a risk)? The second is to evaluate the issue and
assess its urgency and significance. The next is to assign responsibility for resolving the issue (or redirect it to a different management process if it is determined to be a risk or a crisis). The last action is to resolve the issue. Associated processes may also be required to log, track, report and update the issue in an information system.

Crisis or disaster responses (Unknown-Knowns). The literature indicates that responses to crises and disasters are most effective when they are focused and specific to a particular threat condition. This is one of the major challenges for software-developing organizations to address in developing a threat management capability. However, several generic response actions can be gleaned from the literature. The first is to mitigate the threat, in a similar manner to risk mitigation. That is, to take actions to reduce the likelihood of a crisis or disaster occurring and/or the magnitude of the impact if it does. This can be more difficult for crises than risks due to the greater uncertainty associated with this type of threat. One approach is similar to the use of analytical frameworks in risk management to identify categories of risk rather than individual risk factors. For crises, this involves identifying the main categories of crises that might impact the organization’s software engineering capability and/or critical software projects, and develop mitigation and response plans for each category. This, for example, is essentially the approach taken in disaster recovery planning (DRP) for the ‘loss of systems infrastructure’ threat category.

The second response is to prepare for crises and disasters before they arise. This relates to putting appropriate structures and processes in place to coordinate resources and activities in response to a specific crisis. For example, [31] suggests creating a crisis management office within the organization’s governance framework, charged with mitigation, as described above, scanning for crisis warning signals, and to serve as the loci for developing response capabilities. Preparation might also include establishing procedures for rapid mobilization of crisis response teams for critical crisis categories. The other two crisis responses in Figure 3, respond and repair and/or recover, are then be coordinated within this framework.

Responses to unknown threats (Unknown-Unknowns). These responses account for the supra threat management processes described in the top right hand quadrant of Figure 2. Some are common to the other processes. Create contingencies (plans and capabilities) involves developing contingency plans and/or organizational capabilities as described for risk and crisis management. Both are critical. As argued in the literature, threat management is an intricate combination of knowing what to do in the face of uncertainty and surprise, knowing how to do it, and having the necessary capability to do it. Creating contingencies is about all three. Similarly, developing the capability to detect threats, whether issues, risks or crises, is also an important organizational competency. Not all threats can be identified before they impact. However, the literature on socio-technical disasters consistently indicates that a series of warning signals are often evident but ignored before disaster eventually strikes. Capabilities to read these signals and intervene early are critical to averting negative impacts. Finally, assess and assign are similar actions to those described above for issues. Assessment includes analyzing the threat to determine its significance and how it might best be handled before assigning it to one of the other processes (risk, crisis or issue) to be managed away or towards zero impact.

4. CASE ILLUSTRATIONS
This section describes some case examples of organizations at various stages of moving beyond traditional risk management to managing software project threats.

- **Web publishing.** A government agency responsible for publishing final year school exam results was exposed to an extreme risk of security invasion through its web site. It was common practice for students to attempt to breach security and invade the site before the publication date to access marks early. Each year the agency thoroughly planned and prepared for the threat, taking every reasonable security design option and mitigation action, and predefining contingency plans and responses in the event that a breach occurred. A security breach did occur, but in an unexpected way. However, through well-rehearsed procedures, the breach was sealed off within minutes of it occurring and being detected, and a major leak was averted.

  This incident demonstrates an informal but effective form of management to contain a threat that was not identifiable until after it impacted the project. The response to the threat utilized a threat management capability that the web publishing team had built up over several years. This response capability was rapidly applied to contain the impact, eliminate the exposure and minimize the damage, ahead of full analysis of the mechanics of the breach. To the uninformed observer, this may have looked like a failure of risk management since a breach did occur. However, given the state of preparedness of the agency, this is an unreasonable conclusion because it was unlikely that every possible threat could have been foreseen and prevented before it was realized.

- **EFTPOS rollout.** Over several years, an Australian DMV developed a semi-formal practice of having a dedicated response team of technical specialists on standby whenever major new systems functionality was rolled out across its 100 plus office locations. Led by the technical development manager, the team comprised system administrators, database administrators, software engineers responsible for the design of major changes in the release, the help desk manager, and production support specialists. The team leader would monitor calls coming into the help desk at the start of business after a new release to look for indicators of major issues or trends across multiple offices. If a major problem was identified, the team would be rapidly deployed to respond to the problem.

  The response was tailored to the nature of the problem. Typically, it would begin with isolating the problem and putting a ‘workaround’ in place so that production operations could proceed as much as possible, allowing the team to investigate the problem and its cause in detail, and determine and put an appropriate solution in place.

  Such a crisis arose when functionality to accept EFTPOS payments was rolled out. Rather than adding a separate third party EFTPOS service, an integrated solution was developed within the licensing and registration system. However, the ability to test the new function was limited because it
interfaced to live external card services providers. Testing was mostly limited to ‘turning transactions around’ as approved at the EFTPOS gateway rather than processing end-to-end live transactions. When it was released, the service intermittently failed. Worse, some customers were charged twice (first, when the service appeared to fail and then, again, when a retry was successful). The response team immediately deactivated the EFTPOS function and instructed the offices to revert to manual procedures. On investigation, three separate software failures were identified as the cause. Fixes were progressively released and tested at one office until they could be rolled out to all sites.

- **Banking systems.** When a major bank initiated a $400 million project to update its customer facing systems, it recognized the need to extend its usual technology project risk management governance mechanisms with new team-based structures and processes. These teams scanned and monitored threats on the basis of the business services that were to be provided by the new systems and the technical services to be applied in developing them. Still referred to as risk management, the teams met weekly to review known threats and consider any possible new threats from a zero base. They were so successful in keeping the project on track that when the rollout milestone was reached on time, the businesses were caught by surprise because they assumed that such a large undertaking would naturally be late.

## 5. BEYOND RISK MANAGEMENT

For organizations moving beyond risk management to threat management, the following steps will aid in the transition:

1. **Get commitment.** Moving beyond the status quo requires more than the software engineering or IT function to commit to the process. It requires widespread commitment from executive management, business unit managers and sponsors, and the IT function. This makes threat management an IT governance issue. It requires an attitudinal change (to adopt a broader threat-based view of projects) and a willingness to commit resources, structures and processes to manage threats within and across projects.

2. **Agree scope.** Within this commitment, the scale of investment in threat management must be explicitly discussed and agreed, balancing potential benefits and costs for the specific needs of the organization. This will take into account the vulnerability of the organization’s software projects and its risk profile. The management strategy literature suggests that the best way forward in uncertain environments is incremental movement towards desired goals. Agreeing the goals and the pathway to achieving them will reinforce the commitment of stakeholders.

3. **Set up frameworks.** Consistent with the agreed scope, extend existing governance mechanisms by establishing structures to oversee the roll-out and ongoing operation of the planned threat management function, and assign key responsibilities and decision rights. This will include recruiting dedicated resources to staff the function. It may also include creating a project management office-like threat management office, as suggested by [32], and/or integrating responsibilities into existing management committees or capability councils.

4. **Set up processes.** Within these frameworks, start to prepare for threats by identifying generic threat categories and developing and agreeing process models for critical threat management activities and responses within these categories, before building and implementing the processes.

5. **Conduct exercises.** Formal threat management mechanisms are of limited value in themselves, it is the organizational capability to practice threat management in real time, leveraged by other resources, that is the differentiating factor between a crisis-prone and a crisis-prepared organization. This can be similar to and an extension of routine disaster recovery testing exercises, for example. The purpose of these exercises is to practice responding to unforeseen and unexpected disruptions to major projects and discovering the gaps in existing processes and capabilities. The frequency and scale of threat exercises is co-determined with the scope of commitment in step two. Tokenism, as often happens with risk management, is unlikely to show a return on investment.

6. **Be alert.** Sensemaking helps organizations extract order and meaning from equivocal environments [61]. Developing the ability to identify and link together the threads of isolated warnings that might point to the emergence of a threat requires a deliberate investment in learning. Equally important is the ability to identify and initiate an effective response to a rapid onset threat that provides no prior warning. To be effective, threat recognition must be closely linked to the organization’s threat response capabilities.

7. **Review and learn.** As noted in Section 2, organizations find it difficult to learn from threats. They tend to see them as calls for reaction rather than reflection, and believe the same event is unlikely to occur again. As with risk management, integrated threat management is an iterative process of preparation, detection, assessment, assignment and review. In this case, arguably, continuous learning in how to manage threats may be a more sustainable process than learning how to respond to specific events that may rarely be repeated. An example of learning from a non-IT natural disaster is provided by [59].

## 6. CONCLUSION

This paper contributes a theoretical framework of threat management that integrates related issue, risk and crisis management processes to provide a broader spectrum of organizational capability to respond to routine events as well as uncertainty and surprises that threaten software projects and organizational outcomes. The framework is intended to serve as a reference model for software-developing organizations that want to extend their threat management capability beyond traditional boundaries. It can also serve as a benchmark against which to evaluate an organization’s current threat management capability.

This initial conceptual step towards a theoretical framework has limitations. While grounded in the problems and challenges of projects in practice and extant research from other domains, the proposed threat management framework needs formal empirical application and validation in the software project context. Formal statements of propositions and explication of the interrelationship between threat management and project management also need to be developed. These refinements are tasks for future research.
Finally, practical matters of implementation of threat management may also be an area for further consideration. As indicated by the literature review, lethargy in fully embracing risk management is already a problem in many organizations. Moving beyond this level to a broader and deeper commitment to threat management will require a clear set of value propositions and engagement strategy for making the transition. Further case studies will also inform theory on transition paths and behaviors that work in practice.

In this initial form, the proposed integrated threat management framework makes several contributions to research and practice. It refocuses research attention on project threats and provides a framing for viewing and managing these threats in an integrated way. The literature notes that research in risk management has tended to lag the needs of practice. Consequently, the paper also aims to reinvigorate risk-related research to bridge the research-practice gap by extending and improving our understanding of barriers to software project performance and the contribution that threat management can make to improving outcomes.

With respect to the tendency of practice to lag in adopting the prescriptions of research, the proposition made here is that, as with risk management, partial or piecemeal implementation of threat management is unlikely to produce full benefits. Incrementally extending threat management practices to cover the spectrum of certainty and uncertainty of potential negative project impacts will refocus software developers on the potential benefits of improving threat management practices and improving project results beyond what might be ‘good enough’. Practical steps are provided to help organizations make this transition.

The impact of under-performing or failed software projects can be substantial. Not all threats that impact software projects originate in software engineering practices. Indeed, software engineering has contributed many process improvements. However, industry survey data suggests the benefits of these improvements may be lost through projects that, as the vehicle for delivering software-based innovations, only prepare for foreseeable high priority threats. Better threat management may also improve software engineering processes. Researchers and practitioners must continue to develop and apply new management processes to improve the performance track record of software projects. Integrated threat management, as a project and organizational capability, aims to contribute to achieving this outcome.

7. REFERENCES


