



---

# Carnegie Mellon

## 17-653

### Managing Software Development Project Report

---

## Software Project Risk Management: Why Is It Hard and What Can Be Done

### Abstract

There is an abundance of information on software risk analysis and management—methods, processes, frameworks, and assessments—and ample reasons why risk management should be done. In addition, most software development methodologies include a risk management component. Yet most organizations, even those that incorporate risk management, fail to manage risk effectively. One has to look no further than the dismal record of software development: cost overruns, schedule delays and project failures are the norm. Software managers appear to know what to do, but just don't do it.

This paper investigates the factors that prevent software development organizations and their managers from adequately perceiving, analyzing and managing risk, and attempts to discover ways that can make an organization more effective at managing software risk. This paper is not about which process or methodology is “best” but rather it uncovers practical techniques that can make an organization more receptive to the best that a risk management process has to offer. In a nutshell this paper addresses why risk management is hard and what can software managers do about it.

The intended audience consists of those currently involved in software project management as well as senior management in organizations in which software is developed. Software developers and team leaders may also find some of the discussion relevant to their work.

Author:	Chris Lord
Copyright:	Copyright © 2002 by Chris Lord. All Rights Reserved.
Document Version:	0.0.001
Edit Number:	148
Last Revised:	12/3/2002 3:13 PM by Chris Lord
Date Printed:	12/3/2002 3:13 PM

---

*Man prefers to believe that which he prefers to be true.*

*If a man will begin with certainties, he shall end in doubts,  
but if he will be content to begin with doubts, he shall end in certainties.*

*The human understanding, from its peculiar nature,  
easily supposes a greater degree of order and regularity in things than it really finds.*

[Francis Bacon, 1561-1626]

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	Risk Management .....	1
1.2	Process Maturity Models .....	3
1.3	Spiral Development Model.....	4
1.4	Importance of Risk Management .....	4
<b>2</b>	<b>Why Risk Management is Hard.....</b>	<b>4</b>
2.1	Insufficient or Deficient Process .....	4
2.2	Uncertainty and Error .....	5
2.3	Risk Aversion and Arrogance .....	6
2.4	More than a Process.....	7
2.5	More than a Project .....	8
<b>3</b>	<b>What Can Be Done .....</b>	<b>8</b>
3.1	Institute a Process .....	8
3.2	Assign a Risk Officer .....	9
3.3	Generate Information Pull .....	9
3.4	Rewards and Incentives.....	10
3.5	Organizational Context.....	11
<b>4</b>	<b>Conclusion.....</b>	<b>11</b>
	<b>References.....</b>	<b>12</b>

## Executive Summary

Software development must be hard, as evidenced by its dismal historical performance. Cost overruns, schedule delays and project failures are the norm. Despite an abundance of information on risk analysis and management and ample reasons why it should be done, software managers still fail to adequately manage the high-risk activity of developing software. Software risk management must also be hard. Some factors that contribute to this difficulty include:

- *Deficient software development processes.* Software development processes, such as project planning, estimating and tracking, are a necessary precursor of risk management. Without them, managers may substitute intuition because of past uncertainty in forecasts.
- *Risk-averse or reckless attitudes.* A risk-averse culture inhibits risk management since it generally rewards crisis management and heroics. Success is based upon minimizing the thought of possible failure rather than the risks. A manager may deliberately ignore risks to project a confident ‘can-do’ attitude and avoid the semblance of defeatism.
- *Difficulty estimating and managing uncertainty.* Estimates are subject to many sources of error, even when formal methods are used. The uncertainty is often forgotten once estimates are committed to paper. Padding of estimates compounds the problem and exposes schedules to moral hazard, which causes work to expand to fill all available time.
- *Failure to consider organizational context and unique risk factors.* Risks define the boundary conditions and are an integral part of strategy. Many risk factors are not on standard risk analysis checklists; they are unique to the project context and organization.

While there are many possible steps to improve risk management in an organization, some that appear useful and that are often overlooked in traditional risk management include:

- *Create the role of a risk officer.* This person is responsible for identifying, assessing and tracking the things that can go wrong. Such a separate position avoids the potential conflict with those who desperately want a project to succeed.
- *Train managers to elicit risk information through improved communication methods.* Those directly working on a project are in the best position to recognize risks, but often are unwilling to communicate them. One means of creating pull for such information is to condition people at all levels to ask questions that elicit risk causes and characteristics.
- *Align rewards with risk management activities.* Rewards flow to the heroes in a crisis. They must also accrue to those who identify and manage risks early, even if those risks become problems. Heroes are not just problem solvers; they are also problem avoiders.
- *Examine risks in the context of the organization.* No framework or checklist can account for the risks present in all projects. Organizational context must be considered in conducting risk analysis with attention to identifying factors unique to the project.
- *Manage risks across an organization.* Contingency budgets are often considered only in the context of a single project but should instead be managed across a portfolio of projects at the organizational or corporate level. This also limits moral hazard exposure.

These techniques are not typically part of risk management processes. And therein lies perhaps the most important lesson: risk management is more than a process; it requires the right attitudes and the right behavior. Risk management may be hard, but it doesn’t have to be.

# 1 Introduction

Software development is hard. That would seem to be the conclusion from the dismal performance record in the history of software development. Consider a 1979 study of 23 large software development projects in diverse application areas that found 48% were late or over budget by more than 20% [ASA79]. The overwhelming reason given for the failure of these projects was managerial misjudgment: incorrect estimates, inadequate study and insufficient fact finding. The balance was attributed to organizational factors such as lack of leadership, development inexperience and the reallocation of resources. An empirical study ten years later of the causes of software project delays found that most factors were still managerial or organizational and not, as often suspected, due to task complexity or technology problems [Van91]. Twenty years after the 1979 study, a survey by the Standish Group of 7,500 large corporate software development projects eerily recalled similar results: 46% were significantly late or over budget, and 28% failed completely [Pol99]. And the reasons to this very day haven't changed: software development organizations still produce unrealistic schedules and inaccurate resource estimates, and they fail to manage requirements and quality [Hum02]. Organizations fail to manage uncertainty and they fail to manage risk.

Software risk management is hard. Despite an abundance of information on software risk analysis and management—methods, processes, frameworks, and assessments—and ample reasons why it should be done, organizations still fail to adequately manage the high-risk activity of developing software. This paper investigates some of the reasons software development organizations do not adequately perceive, analyze or manage risk, and presents some approaches that can help organizations manage risk more effectively.

## 1.1 Risk Management

Risk is the probability of realizing unwanted consequences of an event or decision [Cha89, p. 51]. The definition of risk includes an element of uncertainty, an element of impact and an element of choice. Risk exists due to uncertainty in time, control and information, but even armed with complete information and unlimited control, there would still be risk. The choices made to control one risk can conflict with others. Consider, for example, the decision to reduce schedule risk that simultaneously increases the risk of a competitor being first to deliver a product to market. Impact determination is based on whether such consequences are acceptable in light of the project goals.

This is the basis of risk management: to assess the probability of experiencing an adverse outcome, to assess the magnitude of loss that would be suffered, and to respond proportionately to reduce the probability or magnitude. The purpose is to improve the chances of a project's success by controlling the potential contributors to failure.

There are several frameworks which formalize the activities and outcomes of risk management. Boehm, for example, groups steps of risk management into assessment activities and control activities [Boe91]. Risk analysis identifies potential problem areas, quantifies the risk exposure of these areas and determines the most significant issues to address based on exposure. Risk control identifies actions that can be taken to mitigate risks, selects from among these alternatives which to implement, and monitors the outcome as feedback for further risk

management. This framework is illustrated in Figure 1-1. The Software Engineering Institute’s Software Risk Evaluation method and Continuous Risk Management methodology both share similar features [HH96].

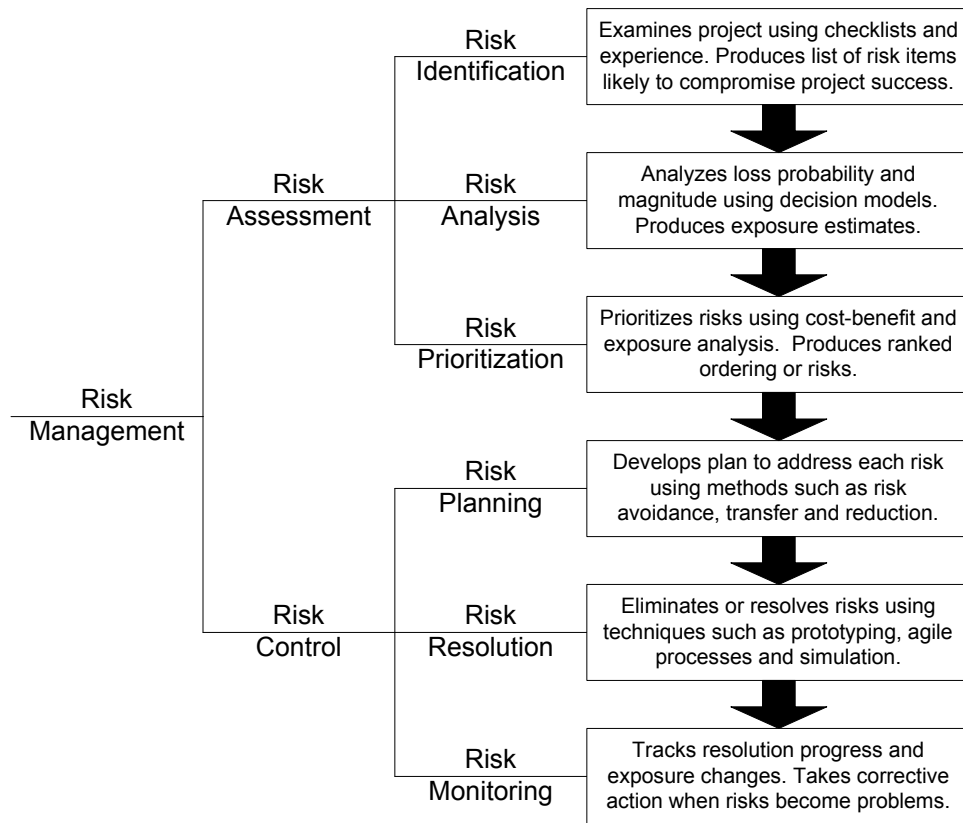


Figure 1-1: Risk Management Activities

In an effort to jump-start organizations, Boehm provides a list of the top ten risk factors and possible management techniques, based on a survey of software project managers. These are listed in the table below.

Risk Item	Risk Management Technique
Personnel Shortfalls	Staffing with top talent; job matching; team-building; morale-building; cross-training; prescheduling key people.
Unrealistic Schedules and Budgets	Detailed, multisource cost and schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing.
Developing the wrong software functions	Organizational analysis; mission analysis; operational concept formulation; user surveys; prototyping; early users’ manuals.
Developing the wrong user interface	Prototyping; scenarios; task analysis.
Gold-plating. Requirements scrubbing	Prototyping; cost-benefit analysis; design to cost.
Continuing stream of requirements changes	High change threshold; information-hiding; incremental development (defer changes to later increments).

Risk Item	Risk Management Technique
Shortfalls in externally-performed tasks	Reference-checking; pre-award audits; award-fee contracts; competitive design or prototyping; team-building.
Shortfalls in externally-furnished components	Benchmarking; inspections; reference checking; compatibility analysis.
Real-time performance shortfalls	Simulation; benchmarking; modeling; prototyping; instrumentation; tuning.
Straining computer science capabilities	Technical analysis; cost-benefit analysis; prototyping; reference checking.

From "Software Risk Management: Principles and Practices" by Barry Boehm in *IEEE Software*, Jan 1991

These and many other strategies for performing risk management are well researched and documented. Even with such information at their disposal, organizations still fail to manage risk.

## 1.2 Process Maturity Models

A process maturity model characterizes the ability of an organization to develop and deliver quality software. It looks at the types of processes that are used within an organization, how well they can be repeated from project to project, and the capacity for improvement and adoption of new technologies and processes. Of the several models available, the Software Engineering Institute's Capability Maturity Model (CMM) is one of the most common and the benchmark against which other models are compared. The CMM software process assessment framework consists of five levels:

<i>Level 1</i> <i>Initial Process</i>	An organization's work is unplanned and uncontrolled. It depends primarily on individual efforts. This level is often characterized by ad-hoc crisis-driven methods.
<i>Level 2</i> <i>Repeatable Process</i>	Planning, tracking, configuration management and quality controls are in place such that a manager can predict development with reasonable accuracy. The process is still dependent on individuals.
<i>Level 3</i> <i>Defined Process</i>	An organization has standardized development activities and established an engineering process group to oversee the formalization of practices. Development processes are considered institutionalized and independent of individuals.
<i>Level 4</i> <i>Managed Process</i>	An organization has instituted data gathering and analysis processes which are used to evaluate product quality and development effectiveness.
<i>Level 5</i> <i>Optimized Process</i>	An organization has mechanisms to introduce new technology into software development and can manage ongoing improvements.

CMM is important in the context of risk management because each maturity level is really a measure of the level of risk that a software development organization is unable to control [Mye96, p. 81]. Organizations at higher maturity levels have processes that reduce the risk of developing software.

This should not be confused with risk management itself. Processes that control and reduce risks are a necessary part of risk management, but are not a substitute for it. The maturity level identifies whether the organization has the infrastructure and processes—such as the practice of software estimation and requirements management—to perform risk management. The practice of risk management is a separate CMM process.

### **1.3 Spiral Development Model**

The spiral development model [Boe88] is a risk-driven meta-process model that guides software development through four main phases, which include determining objectives and constraints, identifying and evaluating risks, and resolving risks. Its distinguishing features include a cyclic approach for incrementally growing a system's degree of definition and implementation, and a set of milestones for ensuring feasibility of the incremental definitions and implementations [Boe00]. Boehm's spiral development model was the first to make risk management a formal software engineering activity [Cha96].

### **1.4 Importance of Risk Management**

Risk Management is important. The Airlie Software Council, a gathering of software development experts at the behest of the Department of Defense in 1995, identified nine “best practices” for software development. The most important of these was formal risk management [You96, p. 133]. The council recommended a number of practices that should be part of any risk management discipline and are consistent with the practices of Boehm and the SEI.

## **2 Why Risk Management is Hard**

Without even attempting a risk management process, the reasons why such a process might be difficult are moot. The issues discussed in the following sections are specifically concerned with the difficulties and challenges encountered by organizations and their managers in implementing risk management. The first step, then, is to implement a risk management process if one is not already in place. For organizations that are in this early stage, an awareness of the following issues will both reinforce the importance of risk management and help guide them among the many challenges.

### **2.1 Insufficient or Deficient Process**

Software development processes are an important component of risk management and, in most cases, a necessary precursor. Risk management requires the project planning and tracking processes associated with a CMM process maturity of level 2. Without this base level, the organization is undisciplined. In commenting about the main causes of software project failure, Humphrey observes:

First, why do competent software professionals agree to completion dates when they have no idea how to meet them? Second, why do rational executives accept schedule commitments when the engineers offer no evidence that they can meet those commitments? Where software is concerned, many otherwise hardheaded executives willingly accept vague promises and incomplete plans. Management's undisciplined approach to schedule commitments contributes to every one of the five most common causes of project failure. [Hum02]

This madness is typical of organizations at the lowest CMM process maturity level. The success or failure of projects in a level 1 organization is not dependent on the nature of the process being followed, or even the skill of the managers. It depends entirely on the skill and qualities of the individuals on the project. Employees move from crisis to crisis. The cost, schedule and quality are unpredictable. [You96, p. 23-24]

A level 2 maturity level is not a sufficient solution either. Organizations think that effective risk management will follow from a repeatable process and widespread access to adequate information about risk management. “Following a repeatable process may mean we are just systematically managing risk poorly. Likewise, having adequate sources of knowledge doesn’t necessarily motivate people to use them correctly. [Gem97]”

A second process-related problem is the failure to effectively use risk management processes that may be in place. This can easily occur in organizations that are in the process of institutionalizing their processes such that some practices are still driven by individuals and will necessarily vary from one individual to another. Without an institutionalized risk management process, risk management won’t happen. This can be seen in a study of corporate risk management where the methods of risk management were well known to managers. Yet “managers made almost no serious attempt to use the common models for risk evaluation. [NE02]” The reasons given were not unlike those seen in software development: managers preferred to rely in their own intuition because of the uncertainty of forecasts or the inputs to models; the belief that risk analysis requires significant time and resources with insufficient return; and lack of familiarity with risk assessment models.

Intuition as substitute for risk assessment has a very poor history, both in software development and in the world at large. Consider, for example, a study in The New England Journal of Medicine that showed that 44 percent of people questioned were prepared to accept a risky treatment if told it would give them a 68 percent chance of surviving. When the risks of the same treatment were presented as a 32 percent chance of dying, only 18 percent said they would undergo the treatment [Pas89]. Does the project have a 68% chance of being on schedule, or a 32% chance of being late? How the impact and probability of a risk is framed has a significant effect on how it is perceived. The risk assessment component of risk management helps counter intuition bias by normalizing risks via risk exposure calculations and forcing risks to be considered relative to other risks.

## 2.2 Uncertainty and Error

Managing software development requires software estimation using methods, models, working practices and past experience to assess the amount of effort required by a project. Clearly if there is no software estimation process, then there will be maximal potential for uncertainty in the forecasts. However, even when managers practice software estimation, “they usually do not handle properly the uncertainty and risks inherent in estimates. [KL97]”

Kitchenham and Linkman describe the four sources of uncertainty that complicate any software estimation. These can be summarized as follows:

*Measurement*      Measurement error is due to limitations on the accuracy of inputs to a model. For example, a study of function point reliability shows that this common input can vary 12% or more across different raters [Kem93].

<i>Model</i>	Model error occurs because models are abstractions that necessarily omit factors present in the situations they model. It also occurs when models have not been adjusted to an organization based on past performance.
<i>Assumption</i>	Assumption error occurs due to inaccuracy in the factors that affect a model's input parameters. For example, incorrectly identifying requirements will result in a function point estimate that does not reflect the true scope of the project.
<i>Scope</i>	Scope error results when an inappropriate model is applied to a project. A model is attuned to a particular situation which limits its applicability.

Of the mistakes the authors described, the most common is a tendency to double-count uncertainty. This occurs because the risk exposure due to assumption error is independent of the model and measurement error. When the impact of a wrong assumption is accounted for in a contingency to the overall estimation, then it should not be accounted for in the model.

The uncertainty in estimates often vanishes once a schedule is created and the numbers put down on paper. Lister's mantra bears repeating: "no project runs according to plan. [Lis97]" Once a schedule is created, people actually believe it will run according to plan. Compounding this reification of estimates is the fact that most "plans account for only the work recognized as absolutely necessary to build the software."

Padding estimates in a schedule is not an effective way to deal with uncertainty or a failure to account for work. Projects grow to consume any available estimate overage in a schedule. Since padding on a schedule represents a form of insurance, this phenomenon can be likened to the well known tendency for insurance to affect behavior, known as moral hazard. The willingness to engage in risky behavior increases with the amount of insurance coverage that limits liability.

## 2.3 Risk Aversion and Arrogance

Two factors that make risk management difficult stem from the attitudes prevalent in organizations—particularly among managers. It is ironic that one of these impediments is risk aversion, which leads to denial. Carr writes:

A risk-averse culture inhibits risk management more than does the lack of a management infrastructure or a repeatable method. Such a culture generally rewards crisis management and punishes those who identify why the project might not succeed. A risk-averse culture relies on heroics to complete a project. When executive management effectively shoots the messenger who bears news of a potential risk, risk management at the project level is doomed to failure. [Car97]

Charette also includes denial as a major reason risk management is not usually done as part of project management: "software project success is based upon minimizing the *thought* of possible failure. [Cha89 p 63]" Typical organizations are focused on the success of a project. Owning up to risks is considered defeatism. "Thus, a manager faced with a nearly impossible schedule may deliberately ignore risks to project a confident, 'can-do' attitude. [BD97]" The problems created by a "can-do" attitude, paradoxically, increase with the exposure and difficulty of a risk:

Fatal risks, on the other hand, offer an entirely different challenge. These risks are either beyond effective mitigation or unacceptably costly to mitigate...Fatal risks are often ignored by the can-do manager. An attitude that would be obviously stupid in the presence of small risks is somehow considered less stupid for large ones. [BD97]

Risks that are obvious and easy to mitigate are apparently handled without any loss of pride. However, those perceived as having the greater consequences elicit a counterproductive heroic attitude of success.

It is not only managers who are subject to such hubris. Yourdon writes that if you're a typical younger developer not only might you be unaware of the risks confronting you, but "you may truly believe that any such risks that might emerge can be overcome, because you're real smart and you're willing to work real hard. [You96, p. 135]"

Many organizations in fact foster such attitudes as part of the can-do ethic. Risk averse and arrogant attitudes lead to software development dominated by crisis management and heroics. The organizational incentives are often structured to reward heroics and "can-do" employees. This positive reinforcement further ingrains these destructive attitudes.

## 2.4 More than a Process

Risk management is more than following the steps of a risk management process. "Software process describes that which is common from project to project...Risk management describes what is different about your project from all others. [Lis97]" No standardized process can capture the particulars of a project. This assertion is reflected in the results of a study of the constructs (a bipolar or scalar comparison metric) that software project managers actually use to assess risks compared with those documented in risk management literature [Moy97]. In particular, it was shown that there are many elicited constructs that did not have equivalents in the SEI risk identification taxonomy [Car93] or among other common risk identification variables. Some of these differences can be attributed to the differences between the types of projects included in the surveys—information systems projects in organizations of less than 4000 people—and the "larger, more formalized, and more technical projects for large industrial and defense organizations" that the SEI risk taxonomy targets.

These omissions may be a direct consequence of differences between the contexts in which the different studies upon which the literature is based were carried out. If this is so, the notion of building a single, all-encompassing risk taxonomy for use by all software developers is probably unrealistic. We may need different risk taxonomies for different project contexts. [Moy97]

Additionally, the construct variation between project managers was also significant despite apparent similarity between projects. This casts serious doubt on the ability for any single checklist to capture the subtleties and variations between projects. Risk identification must be adapted to the project and organizational context. The results and conclusions of this study are also supported in separate studies of 83 software project managers [RL00] and 45 software project managers [KCLS98].

Another common problem with risk management is it is treated as a separate activity with its own reporting, costs and outputs, much as the analysis phase is considered an isolated activity in traditional linear development models. "When thought of as simply a process, risk management seems separate from other activities. It becomes a topic addressed only in a section of a plan or part of a review. [Gem97]" Risk is present in every decision, every issue and every problem and needs to be considered as such. This recognition was the motivation for the spiral lifecycle model, which incorporates risk management activities at every stage.

## 2.5 More than a Project

Risk management is hard when it is performed at the level of an individual project. It is important to have an organization that is amenable to risk management and offers a supportive environment. But it is also in the organization's best interest to perform risk management at a level above the individual projects. Risks define the boundary conditions within which one operates [Cha89, pp. 62-63]. They therefore form an integral component of strategy. As companies have recognized the relationship between risks and strategy, there has been a move towards enterprise risk management. "Estimating risk is much more than a mechanical exercise of coming up with 'the numbers.' Instead, it represents a critical step in determining whether the firm's strategies will either create or destroy shareholder value. [NE02]"

A similar point is made by Kitchenham and Linkman in the case of software: "Uncertainty and risk cannot be managed effectively at the individual project level. These factors must be considered in an organizational context. [KL97]" The processes for software effort estimation are equally likely to result in overestimates as they are underestimates. This estimate inaccuracy must be managed across an entire organization and across all projects, much as insurance companies manage the risk across a portfolio of policies.

## 3 What Can Be Done

Successful risk management is dependent on effective software development processes, but it also depends on having the right attitude and the right behavior. Risk-averse attitudes—often reinforced by existing reward and recognition policies—contribute to substantial unmitigated project risk. However, even constructive attitudes are not sufficient. As Gemmer observes: "We found we were approaching risk management in a similarly zealous manner. We were obsessed with improving the process and had forgotten the desired result. We used the latest methods and tools, but overlooked people's perceptions of risk and their attendant behaviors. [Gem97]"

No list of remedies can be exhaustive and this section does not even purport to come close. Instead, what follows are some key ideas culled from the literature on risk management and software development. These are approaches that may improve the risk management activities of an organization and, at a minimum, warrant consideration by software managers.

### 3.1 Institute a Process

The problem with most literature on risk management is it begins and ends with the process that should be followed. Managers are admonished to go forth and implement the process. As shown in the first part of this paper, it is rarely that simple. However, there is no denying that a risk management process is a necessary beginning. It is a universally acknowledged among software development experts that some form of risk management is better than none. It is also generally recognized that without a formal risk management process, the results are unsatisfactory. Based on his experience with organizations without a formal process, Carr observes:

The risks identified by ad hoc methods are generally the global risks that the organization is willing to accept. Systematic methods are lacking to ensure that all aspects of the project have been examined for risk, and the project is periodically reexamined to identify new risks. [Car97]

If a formal risk management process is not part of the organization's software development process, it should be.

### 3.2 Assign a Risk Officer

The world of business and finance has witnessed the emergence of a chief risk officer [Shi01, p. 254]. This is ideally a member of senior management responsible for identification and measurement of all risks faced by the company. Such a person should not have direct business or revenue responsibility to avoid creating conflicts of interest. Instead, the risk officer serves as the champion to promote risk management throughout the company.

This same approach can be applied to software development organizations. Yourdon refers to this position as the "loyal opposition." A risk officer is tasked with identifying, assessing and tracking all the things that could go wrong when everyone else on the team desperately wants to believe that things will work [You96, pp. 134-135]. The implication is that such a position should not be held by the project manager or anyone else whose superficial interests are at odds with the purpose of risk management. While it may be possible to have one person manage dual sets of responsibilities, it is far safer to avoid the potential conflict of interest altogether.

### 3.3 Generate Information Pull

Communication of risks is one the most challenging tasks in risk management. The people in the position to best recognize many of the risks are typically those working on the project. For a variety of reasons—some previously discussed—these people may not be willing to communicate the risk. It is therefore necessary to create an environment that generates information pull. Employees must learn that identifying a risk doesn't mean it will occur (and ignoring it doesn't mean it won't).

One means of creating information pull is to condition people at all levels to ask questions that elicit risk causes and characteristics. For example, Gemmer describes in [Gem97] how his organization coached managers to consider certain questions in response to cues, some of which are listed in the table below.

If the team says...	Management should ask...
"There's an issue here"	Is the issue an opportunity, a risk or a problem? What uncertainty surrounds the issue? What choices do we have for dealing with it? Are they proactive or reactive? How does the outcome affect expectations? If it's an opportunity, what risks are associated with pursuing it?
"Here are our risks"	Which risks do you want us to hear? Do you need our help with any of these? Do we have consensus on the risk's characteristics? Do we have consensus on the actions to be taken?
"We're taking a calculated risk" or "We assume that..."	What calculations led you to this approach? What are sources of uncertainty? Which are related to time? Which are related to control? Which are related to information? How can they be minimized? What biases may be present in our perception of risk? What will we do if the risk occurs? What evidence, if found, would affect the validity of the assumption?

If the team says...	Management should ask...
“The risk is being worked” or “There’s no change in the risk”	What actions are being taken? Has any requested assistance been provided in a timely fashion? How have the actions affected the risk’s characteristics? What is the trend over time? Are we being proactive enough? Are we trying to mitigate, avoid, transfer, or accept the risk? Do we need to do more or try something else? Are we approaching the time frame (decision point) for any of the risks? What will we do if the risk occurs?
“The risk is reduced” or “The risk went away”	What has changed about the situation? Do we understand the situation better or has the risk changed? Has the probability or impact decreased? Or both? Why? What biases may be present in our perception of the risk? If the probability decreased, what uncertainty remains? If we chose to avoid the risk, what new risks do we face as a result of those choices?
“There are no new risks” or “There are no significant risks”	What techniques were used to look for them? What insignificant (nonreportable) risks are being tracked? What expectations are still unmet? What things are we uncertain about? How has our situation changed? Are things getting better or worse? How many risks have been averted? How many risks became problems?

From “Risk Management: Moving Beyond Process” by Art Gemmer in *IEEE Computer*, May 1997

Another aspect of generating information pull is to ensure there is diversity in sources [Gem97]. Managers should seek out different perspectives and divergent opinions. This is especially important because minority views do not often attain visibility outside the team or group level.

### 3.4 Rewards and Incentives

Most organizations reward individuals who work hard and the heroes in a crisis even more so. On the other hand, these same organizations suffer from the “shoot the messenger” syndrome. The messenger, in this context, comes bearing news of identified risks. In order to improve risk management, an organization must reward those who identify and manage risks early, even if those risks later become problems. “Heroes are not just problem solvers; they are also problem avoiders. [Gem97]” In order to determine the types of behavior that should be rewarded, Gemmer describes two modes of organizational operation:

*Work-Hard* This mode is characterized by extended overtime, crisis meetings, unforeseen tasks, missed schedule commitments and rework. It may have the outward appearance of an unplanned project even though project planning has ostensibly been done. Work-hard organizations spend most of their time dealing with the consequences of past decisions; they are reactive.

*Work-Smart* This mode has fewer crises, fewer surprises and greater choice. Organizations in this mode spend most of their time considering future options. They have greater flexibility and can anticipate and adapt to changes easily. They are proactive.

The structure, form and frequency of rewards are subjects of debate. Should they be infrequent but large or less frequent and small? It is the responsibility of the manager to know the members of the organization and to offer rewards that are appropriate and personalized. The size of the rewards should be proportional to the value contributed to the organization. Traditionally this is based on the amount of work. An individual incentive for risk management could be based instead on early identification and management: it could reward the process, not necessarily the outcome. Groups should also have incentives. Like individual rewards, they should be based on

work-smart behavior. Small, non-monetary rewards may take the form of t-shirts, mugs, plaques or other form of recognition. They should be provided frequently, particularly at project milestones [Joh99].

### 3.5 Organizational Context

Three studies of software project managers ([Moy97], [Kei98] and [RL00]) all show that no framework can account for the risks present in all projects. These studies compared identified risk factors with existing risk analysis frameworks and checklists. The results underscore the importance of organizational context in conducting risk analysis. The risk factors included those that show up on most checklists, such as Boehm's top 10 risk items [Boe91] and the SEI software risk taxonomy [Car93].

One of the most interesting findings from [Kei98] is the fact that the risks perceived to be most important often lie outside the direct control of the project manager. Most project managers surveyed reported their perceptions of risk were higher for those items over which they had little or no control. This is consistent with psychological studies of attitudes towards risks that show that risks for which a person has less knowledge and which are more dreaded are perceived as the greatest threats and deserving of the greatest attention [Kun02].

Many risk factor checklists focus on what the authors term "execution" risks over which the project manager has a relatively high degree of control. One conclusion is that managers should look at a broader set of factors and not restrict themselves to things they control. To properly manage these larger factors requires the participation and support of the larger organization.

Risk management strategies often suggest a contingency budget be set aside to deal with foreseen problems. This strategy is often considered only in the context of a single project. Since an estimate is a prediction that is equally likely to vary above or below the actual result [KL97], contingency budgets should instead be managed across a portfolio of projects at the organizational or corporate level. The goal of the organization is to obtain an amount of risk consistent with organizational risk sensitivity. The risk levels of individual projects may, however, vary dramatically much as the individual instruments of an investment portfolio may vary widely. A contingency budget would be allocated as needed to maintain the overall desired level of portfolio risk. Another advantage of making the contingency budget a shared organizational resource is the mitigation of the moral hazard of having per-project contingencies: it makes it that much more difficult to access and raises the level of visibility of project overruns and helps improve accountability across the organization.

## 4 Conclusion

Risk management is a vital part of successful software management. Although most software managers know what to do, they just don't do it [KL97]. Some of the factors that contribute to this behavior include deficient software development processes, failure to adopt a risk management process, risk-averse or reckless attitudes, difficulty estimating and managing uncertainty, and failure to consider organizational context and unique project risk factors.

While there are many possible steps to improve risk management in an organization, some of those that appear to have the most potential for success include creating the role of a risk officer

or manager, training managers to elicit risk information through improved communication methods, aligning rewards and incentives with risk management activities, examining risks in the context of the organization, and managing risks across an organization. These techniques are not typically part of risk management processes. And therein lies perhaps the most important lesson: risk management is more than a process; it requires the right attitudes and the right behavior. Risk management may be hard, but it doesn't have to be.

## References

- [ASA79] Abe, J., Sakamura, K., and Aiso, H., "An Analysis of Software Project Failure," *Proceedings of the 4<sup>th</sup> IEEE Software Engineering Conference*, September 1979.
- [BD97] Boehm, B. and DeMarco, T. "Software Risk Management," *IEEE Software*, May/June 1997, pp. 17-19.
- [Boe00] Boehm, B., "Spiral Development: Experience, Principles, and Refinements," *Special Report CMU/SEI-2000-SR-008*, July 2000.
- [Boe88] Boehm, B., "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988, pp. 61-72.
- [Boe91] Boehm, B., "Software Risk Management: Principles and Practices," *IEEE Software*, Volume 8, Issue 1, Jan 1991, pp. 32-41.
- [BS00] Boehm, B. and Sullivan, K., "Software Economics," *Proceedings of the future of Software Engineering Conference*, Limerick, Ireland, 2000, pp. 319-343
- [Car93] Carr, M., et al, "Taxonomy-Based Risk Identification," *Technical Report CMU/SEI-93-TR-6*, June 1993.
- [Car97] Carr, M., "Risk Management May Not Be for Everyone," *IEEE Software*, May/June 1997, pp. 21-24.
- [Cha89] Charette, R., *Software Engineering Risk Analysis and Management*, McGraw-Hill Company, 1989.
- [Col98] *The Columbia Dictionary of Quotations*, Columbia University Press, 1998.
- [Gem97] Gemmer, A., "Risk Management: Moving Beyond Process," *IEEE Computer*, May 1997, pp. 33-43.
- [Han00] Hansen, W., et al, "Spiral Development - Building the Culture," *Technical Report CMU/SEI-2000-SR-006*, July 2000.
- [HH96] Higuera, R. and Haimes, Y., "Software Risk Management," *Technical Report CMU/SEI-96-TR-012*, June 1996.
- [Hum02] Humphrey, W., "Why Projects Fail", *Computerworld*, May 20, 2002.
- [Joh99] Johnson, L., "Herding Cats: Keep Your Cats at Home," *Enterprise Development*, March 1999.
- [Kei98] Keil, M., et al "A Framework for Identifying Software Project Risks," *Communications of the ACM*, Vol. 41, No. 11, November 1998.
- [Kem93] Kemerer, C., "Reliability of Function Points Measurement: A Field Experiment," *Communications of the ACM*, Volume 36 Issue 2, 1993, pp. 85-97.
- [KL97] Kitchenham, B. and Linkman, S., "Estimates, Uncertainty and Risk," *IEEE Software*, May/June 1997, pp. 69-74.
- [Kun02] Kunreuther, H., *Risk Analysis and Risk Management in an Uncertain World*, Working Paper, Wharton, August 2002.

- [Lis97] Lister, T. "Risk Management is Project Management for Adults," *IEEE Software*, May /June 1997, pp. 20-22.
- [Mc96] McConnell, S., *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, 1996, p. 252.
- [Moy97] Moynihan, T., "How Experienced Project Managers Assess Risk," *IEEE Software*, May/June 1997, pp. 35-41.
- [Mye96] Myerson, M., *Risk Management Processes for Software Engineering Models*, Artech House, Inc., 1996.
- [NE02] Noy, E. and Ellis, S., "Risk: A Neglected Component of Strategy Formulation," *Working Paper*, [http://recanati.tau.ac.il/faculty/ellis\\_shmuel.htm](http://recanati.tau.ac.il/faculty/ellis_shmuel.htm), 2002.
- [Pas89] Passell, P., "The Risk Debate: The American Sense of Peril," *Manhattan Institute Civil Justice Memo*, [http://www.manhattan-institute.org/html/cjm\\_14.htm](http://www.manhattan-institute.org/html/cjm_14.htm), May 14, 1989.
- [Pol99] Pollack, A., "For Coders, a Code of Conduct," *New York Times*, May 3, 1999.
- [RL00] Ropponen, J. and Lyytinen, K., "Components of Software Development Risk," *IEEE Transactions of Software Engineering*, February 2000, pp. 98-112.
- [Shi01] Shimpi, P., *Integrating Corporate Risk Management*, Texere LLC, 2001.
- [Van91] van Genuchten, M., "Why is software late? An empirical study of reasons for delay in software development," *IEEE Transactions on Software Engineering*, Volume 17 Issue 6, June 1991, pp. 582 -590.
- [You96] Yourdon, E., *Rise and Resurrection of the American Programmer*, Prentice-Hall, 1996.

It is often difficult to assess the significance of references in a report of this type. Some references are cited for supporting studies or background information, others for their topical relevance and still others to defend the claim of in-depth research. For readers interested in the topic discussed in this paper will find the following references useful:

1. Gemmer, A., "Risk Management: Moving Beyond Process," *IEEE Computer*, May 1997, pp. 33-43.
2. Boehm, B. and Sullivan, K., "Software Economics," *Proceedings of the future of Software Engineering Conference, Limerick, Ireland, 2000*, pp. 319-343.
3. Boehm, B., "Software Risk Management: Principles and Practices," *IEEE Software*, Volume 8, Issue 1, Jan 1991, pp. 32-41.
4. Yourdon, E., *Rise and Resurrection of the American Programmer*, Prentice-Hall, 1996.

In addition, the May 1997 issue of *IEEE Software* contains many excellent articles on risk management by such respected authorities and practitioners as Barry Boehm, Marvin Carr, Tom DeMarco and Robert Charette. This issue should be enjoyed in its entirety.