Risky business:

what we have yet to learn about software risk management

Shari Lawrence Pfleeger

Systems/Software, Inc.,Washington, DC

Journal of Systems and Software, 53, 2000, pp 265-273.

Abstract: This paper examines the way in which computer scientists perform risk management, and compares it with risk management in other fields. We find that there are three major problems with risk management: false precision, bad science, and the confusion of facts with values. All of these problems can lead to bad decisions, all in the guise of more objective decision-making. But we can learn from these problems and improve the way we do risk management.

Keywords: risk, risk management, risk analysis, values

It's in the textbooks, the process models, and the standards: software risk management is essential to developing good products within budget and schedule constraints. So why do we seldom do it? And why, when we do it, do we have little confidence in its ability to help us plan for the problems that might arise? In this article, I explore the state of software risk management today, and investigate what we can learn from other disciplines that have made risk management a required step in their decision-making process.

A risk is an unwanted event that has negative consequences. Ideally, we would like to determine whether any unwelcome events are likely to occur during development or maintenance. Then, we can make plans to avoid these events or, if they are inevitable, minimize their negative consequences. We use risk management techniques to understand and control the risks on our projects.

1. What's wrong with this picture?

To see that there is room for improvement, consider the flaw in the Pentium chip, reported in 1994. At the time the flaw was acknowledged, six million personal computers relied on the flawed chip. At $300 per chip, Intel's risk impact was $1.8 billion, which includes not only the 3-4 million PCs already sold but also the remainder in stores and warehouses. (Markoff 1994) Intel's risk assessment showed that "average" computer users would get a wrong answer (due to the chip's flaw) every 27,000 years of normal computer use, and a "heavy user" would see a problem once every 270 years. Thus, Intel decided that the flaw was not meaningful to most users.

However, IBM performed its own risk assessment and generated very different numbers. According to IBM, the Pentium could cause a problem every 24 days for average users. (IBM 1994) William R. Pulleybank, mathematical sciences director at IBM's Watson Lab, suggested that a large company using 500 Pentium-based PCs could experience up to 20 problems a day! So IBM's assessment was 400,000 times worse than Intel had calculated, and IBM halted sales of its computers using this chip. An independent assessment by Prof. Vaughan Pratt of Stanford University also doubted Intel's numbers; Pratt found an error rate "significantly higher than what Intel had reported." (Lewis 1994)

Lest you think that only computer scientists have trouble with risk management, consider the risk assessment performed by several European governments from 1988 to 1990. Each of eleven countries

(Netherlands, Greece, Germany, Great Britain, Spain, France, Belgium, Italy, Denmark, Finland and Luxembourg) plus several private firms (such as Rohm and Hass, Battelle, Solvay and Fiat) built a team of its best experts and gave it a well-described problem about well-known elements: evaluating the risk of an accident at a small ammonia storage plant. The eleven national teams varied in their assessments by a factor of 25,000, reaching wildly different conclusions. (Commission of the European Communities 1991) Not only did their numbers differ, but many of their assumptions and models differed, including:

- what kinds of accidents to study
- the plume behavior after the ammonia was released
- the consequences of ammonia's entering the environment
- the rapidity of the emergency team's response
- the probability of success of mitigation measures

The commission noted that, "at any step of a risk analysis, many assumptions are introduced by the analyst. It must be recognized that the numerical results are strongly dependent on these assumptions."
So the bad news is that, even on a seemingly-well-understood problem, using science that has been around far longer than software engineering, we are not particularly good at articulating and evaluating our risk. The good news is that we can try to learn from the experiences of other disciplines.

2. What is risk management?

To understand how to improve our risk assessment expertise, we must first investigate how we are being told to evaluate risk today. We begin by asking how to determine what these risks are. Guidance is provided in many places: books, articles, tutorials, and tools, for instance. Most advice asks us to distinguish risks from other project events by looking for three things: (Rook 1993 and Pfleeger 1998)

1. *A loss associated with the event.* The event must create a situation where something negative happens to the project: a loss of time, quality, money, control, understanding, and so on. For example, if requirements change dramatically after the design is done, then the project can suffer from loss of control and understanding if the new requirements are for functions or features with which the design team is unfamiliar. And a radical change in requirements is likely to lead to losses of time and money if the design is not flexible enough to be changed quickly and easily. The loss associated with a risk is called the risk impact.

2. *The likelihood that the event will occur.* We must have some idea of the probability that the event will occur. For example, suppose a project is being developed on one machine and will be ported to another when the system is fully tested. If the second machine is a new model to be delivered by the vendor, we must estimate the likelihood that it will not be ready on time. The likelihood of the risk, measured from 0 (impossible) to 1 (certainty) is called the risk probability. When the risk probability is 1, then the risk is called a problem, since it is certain to happen.

3. *The degree to which we can change the outcome.* For each risk, we must determine what we can do to minimize or avoid the impact of the event. Risk control involves a set of actions taken to reduce or eliminate a risk. For example, if the requirements may change after design, we can minimize the impact of the change by creating a flexible design. If the second machine is not ready when the software is tested, we may be able to identify other models or brands that have the same functionality and performance and can run our new software until the new model is delivered.

We can quantify the effects of the risks we identify by multiplying the risk impact by the risk probability, to yield the risk exposure. For example, if the likelihood that the requirements will change after design is .3, and the cost to redesign to new requirements is $50,000, then the risk exposure is $15,000. Clearly, the

risk probability can change over time, as can the impact, so part of our job is to track these values over time, and plan for the events accordingly.

There are two major sources of risk: generic risks and project-specific risks.

i. Generic risks are those common to all software projects, such as misunderstanding the requirements, losing key personnel, or allowing insufficient time for testing.

ii. Project-specific risks are threats that result from the particular vulnerabilities of the given project. For example, a vendor may be promising network software by a particular date, but there is some risk that the network software will not be ready on time.

We can also think of two types of risks, voluntary and involuntary, depending on whether we have control over it. Involuntary risks, like the risk of cancer from a hole in the ozone layer, are risks we face. In the software realm, we may take involuntary risks by using a new operating system or development platform. On the other hand, we take voluntary risks, such as using inexperienced people on a new project, just as we choose to eat fatty foods even though we are aware of their health risks.

3. Risk management activities

Software engineering textbooks lay out the steps of risk management, often using charts such as the one in Figure 1. First, you assess the risks on your project, so that you understand what may occur during the course of development or maintenance. The assessment consists of three activities: identifying the risks, analyzing them, and assigning priorities to each of them. To identify them, you may use many different techniques.

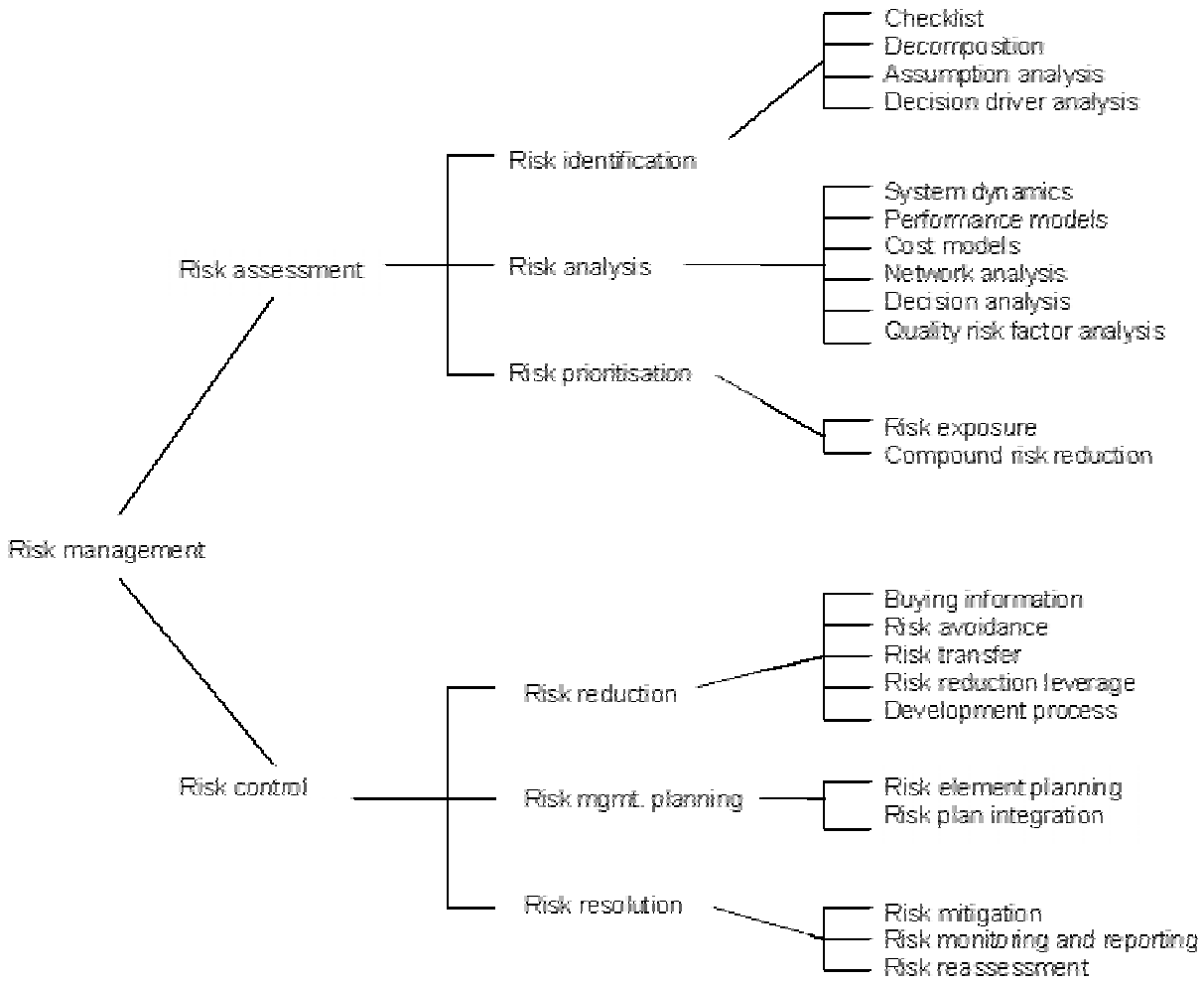

Figure 1. Typical approach to software risk management (from Pfleeger 1998)

If the system you are building is similar in some way to a system you have built before, you may have a checklist of problems that are likely to occur; you can review the checklist to determine if your new project is likely to be subject to the risks listed. For systems that have new characteristics, you may augment the checklist with an analysis of each of the activities in the development cycle; by decomposing the process into small pieces, you may be able to anticipate problems that may arise. For example, you may decide that there is a risk of your chief designer's leaving during the design process.

Similarly, you may analyze the assumptions or decisions you are making about how the project will be done, who will do it, and with what resources. Then, each assumption is assessed to determine the risks involved. You may end up with a list similar to Barry Boehm (1991) who identifies ten risk items, and recommends risk management techniques to address them.

1. **Personnel shortfalls.** Staffing with top talent; job matching; team-building; morale-building; cross-training; pre-scheduling key people.

2. **Unrealistic schedules and budgets.** Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing.

3. **Developing the wrong software functions.** Organizational analysis; mission analysis; operational concept formulation; user surveys; prototyping; early users' manuals.

4. **Developing the wrong user interface.** Prototyping; scenarios; task analysis.

5. **Gold-plating.** Requirements scrubbing; prototyping; cost-benefit analysis; design to cost.

6. **Continuing stream of requirements changes.** High change threshold; information-hiding; incremental development (defer changes to later increments).

7. **Shortfalls in externally-performed tasks.** Reference-checking; pre-award audits; award-fee contracts; competitive design or prototyping; team-building.

8. **Shortfalls in externally-furnished components.** Benchmarking; inspections; reference checking; compatibility analysis.

9. **Real-time performance shortfalls.** Simulation; benchmarking; modeling; prototyping; instrumentation; tuning.

10. **Straining computer science capabilities.** Technical analysis; cost-benefit analysis; prototyping; reference checking.

Finally, you analyze the risks you have identified, so that you can understand as much as possible about when, why and where they might occur. There are many techniques you can use to enhance your understanding, including system dynamics models, cost models, performance models, network analysis, and more.

Now that you have itemized all risks, you must assign priorities to the risks. A priority scheme enables you to devote your limited resources only to the most threatening risks. Usually, priorities are based on the risk exposure, which takes into account not only likely impact but also the probability of occurrence.

The risk exposure is computed from the risk impact and the risk probability, so you must estimate each of these risk aspects. To see how the quantification is done, consider the analysis depicted in Figure 2. Suppose you have analyzed the system development process, and you know you are working under tight deadlines for delivery. You will be building the system in a series of releases, where each release has more functionality than the one that preceded it. Because the system is designed so that functions are relatively independent, you are considering testing only the new functions for a release, and assuming that the existing functions still work as they did before. Thus, you may decide that there are risks associated with not performing regression testing: the assurance that existing functionality still works correctly.

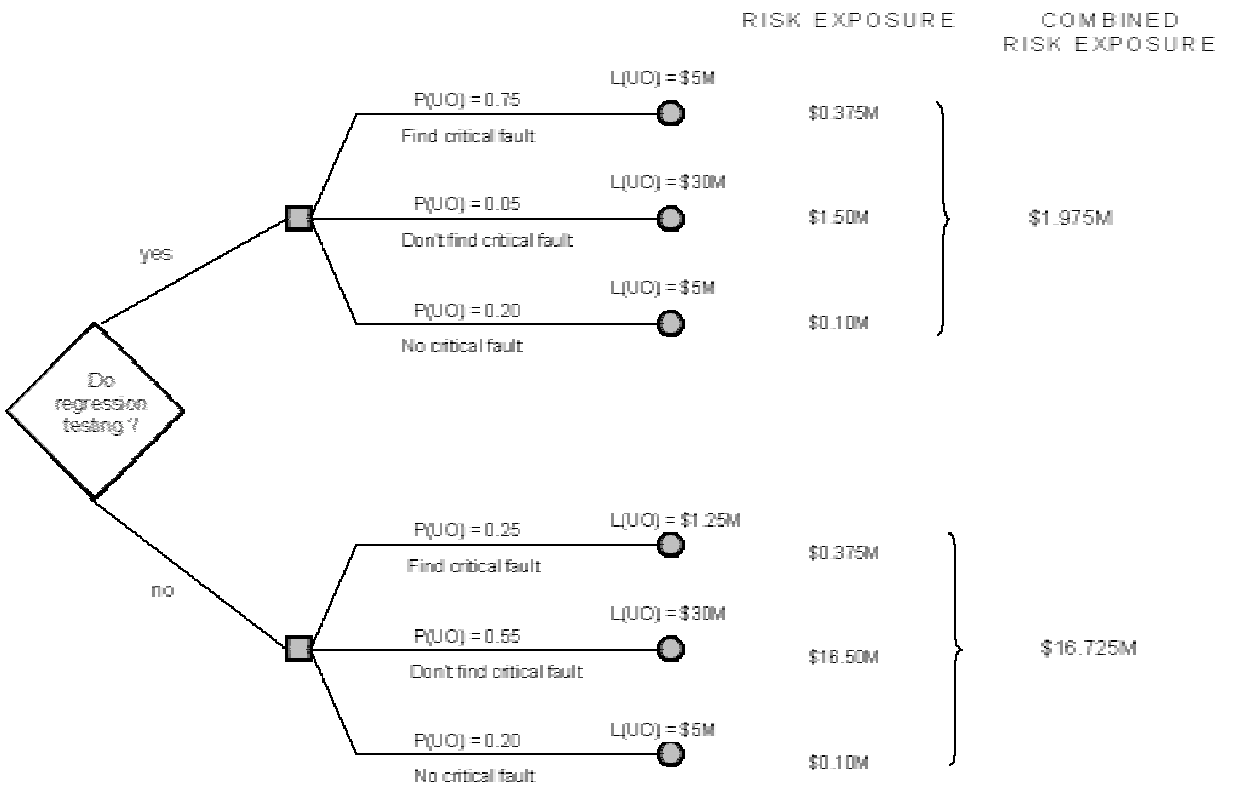

Figure 2. Typical risk calculation (from Pfleeger 1998)

For each possible outcome, you estimate two quantities: the probability of an unwanted outcome, P(UO), and the loss associated with the unwanted outcome, L(UO). For instance, there are three possible consequences of performing regression testing: finding a critical fault if one exists, not finding the critical fault (even though it exists), or deciding (correctly) that there is no critical fault. As the figure illustrates, we have estimated the probability of the first case to be 0.75, of the second to be 0.05, and of the third to be 0.20. The likelihood of an unwanted outcome is estimated to be $.5 million if a critical fault is found, so that the risk exposure is $.375 million. Similarly, we calculate the risk exposure for the other branches of this decision tree, and we find that our risk exposure if we perform regression testing is almost $2 million.However, the same kind of analysis shows us that the risk exposure if we do not perform regression testing is almost $17 million. Thus, we say (loosely) that more is "at risk" if we do not perform regression testing. Risk exposure helps us to list the risks in priority order, with the risks of most concern given the highest priority.

Next, we must take steps to control the risks. We may not be able to eliminate all risks, but we can try to minimize the risk, or mitigate it by taking action to handle the unwanted outcome in an acceptable way. Therefore, risk control involves risk reduction, risk planning and risk resolution.

There are three strategies for risk reduction:

- *avoiding the risk,* by changing requirements for performance or functionality
- *transferring the risk,* by allocating risks to other systems or by buying insurance to cover any financial loss should the risk become a reality

- *assuming the risk,* by accepting it and controlling it with the project's resources

To aid decision-making about risk reduction, we must think about the business value of each risk-related decision, taking into account the cost of reducing the risk. We call risk leverage the difference in risk exposure divided by the cost of reducing the risk. In other words, risk reduction leverage is

(risk exposure before reduction – risk exposure after reduction)/(cost of risk reduction)

If the leverage value is not high enough to justify the action, then we can look for other, less costly or more effective reduction techniques.

Once we have completed our risk management plan, we monitor the project as development progresses, periodically re-evaluating the risks, their probability, and their likely impact.

The steps seem clear and straightforward, but the results do not seem to bear out the promise of risk management. So how do we improve our risk management practices? To help us understand the problems we face, and their possible solutions, we look to other disciplines that use risk management. In particular, the public policy literature offers us a wealth of examples of what is right and wrong about dealing with risk.

4. Avoid false precision

Quantitative risk assessment is becoming more and more popular, both because of its inherent appeal to scientists and because it is often mandated by regulatory agencies. For instance, from 1978 to 1980, only eight chemicals were regulated on the basis of quantitative risk analysis in the US. But from 1981 to 1985, 53 chemicals were regulated that way. Similarly, there are more and more calls for quantitative assessments of software risk.

One of the first things to notice about how the rest of the world handles risk is that most other disciplines consider the probability distribution of the risk, not a point probability. That is, other risk analysts acknowledge that there is a great deal of uncertainty about the probability itself, and that not every possibility is equally likely. Indeed, in 1984, William Ruckelshaus, head of the US Environmental Protection Agency, mandated that the uncertainty surrounding each risk estimate be "expressed as distributions of estimates and not as magic numbers that can be manipulated without regard to what they really mean." (Ruckelshaus, p. 161) One way to improve our success in managing software-related risk is to use distributions, and to base them on historical data, not just on expert judgement.

Another danger of quantifying risk in this way is that the numbers can actually obscure what is really happening. Studies of risk perception reveal that people view a hazard quite broadly. They are concerned about the effects of a hazard, but they also want to know about the hazard's genesis, asking questions such as:

- How would it come about?
- Is it voluntary or involuntary?
- Is it associated with a particular technology?
- Who might be affected?
- Could it be catastrophic?

These characteristics amplify the perception of the hazard, and in turn affect the way the risk is perceived and its probability and exposure are quantified. For example, voluntary hazards are usually viewed with less dread than involuntary ones. That is, we choose to smoke cigarettes, drive on high-speed roads, or eat fatty foods. But we do not choose to have a thinning ozone layer, and we dread the consequences of the radiation even when told that that the threats to our health are less when sunbathing than when eating poorly. Figure 3 illustrates some of these risk characteristics, contributing to discomfort with lack of knowledge and leading to (sometimes unreasonable) dread. Often, risk analysts will place a risk on a grid with two axes, one for each factor, to determine the effects of knowledge and dread on the quantification.

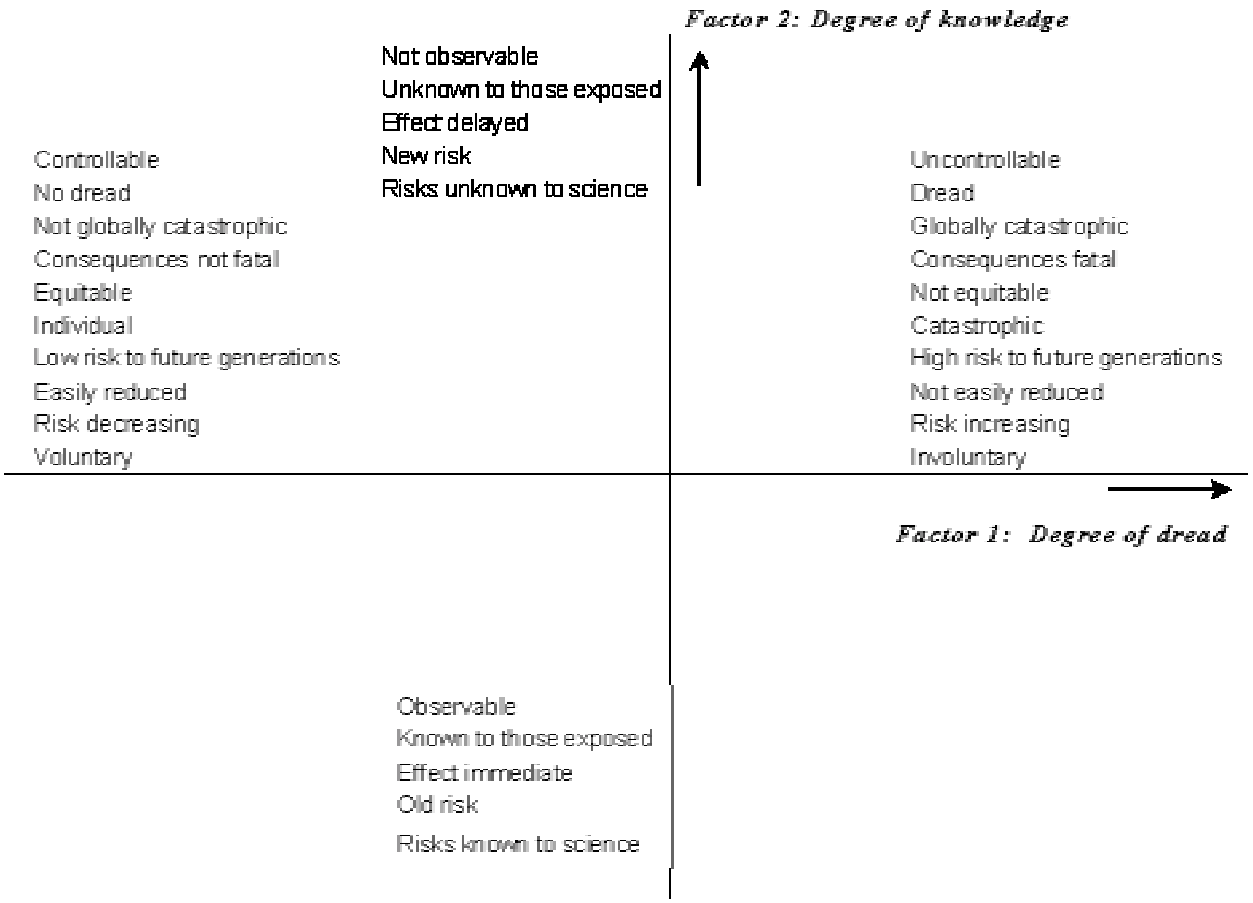

Figure 3. Aspects of risk based on key characteristics (adapted from Slovic, Fishchoff and Lichtenstein 1985)

Other factors (outside science) affecting how risks are assessed include the types of professionals doing the assessment, the composition of committees involved, and the legal and political processes working in conjunction with risk assessment. Thus, who quantifies the risk can be just as important as how the risk is quantified.

5. Don't be fooled by "questionable science"

Related to these issues of quantification and precision are issues of the science used to collect, analyze and present quantified risk information. The most problematic aspect of quantifying risk data is the possibility of misleading regulators and decision-makers into thinking that they can ignore or give less

credence to qualitative data. That is, a numerical description of risk is often given more credence than a qualitative one, even when quantitative descriptions are known to be suspect. Sheila Jasanoff (1991) points out that "numerical assessments possess a kind of symbolic neutrality that is rarely attained by qualitative formulations about the ‘weight’ or ‘sufficiency’ of the evidence." (p.45)

Moreover, error and unreliability are underestimated when sample size is small. Psychological research on risk perception tells us that "both scientists and lay people may underestimate the error and unreliability in small samples of data, particularly when the results are consistent with preconceived, emotion-based beliefs." (Whittemore 1983, p. 28)

A more important concern is our reluctance to examine the study designs that lead to quantitative risk assessments. For example, during the mid-1980s, an expert committee examining pesticide risk found fault with the US Environmental Protection Agency for not paying attention to flaws in the design and conduct of toxicological studies. One committee member described the risk assessment as being based on "gimcrack mathematics." (Jasanoff 1990, p. 136) In the software realm, we are rarely asked to present the studies that support our risk assessments, let alone their underlying designs and data sets. And because we do not always publish or make available the complete set of study documentation, our studies are almost impossible to evaluate or replicate.

One aspect of study design quality that is often ignored is the relevance and quality of the data being studied. We sometimes think of medical studies as the "gold standard" to which we should compare our research. But a random sample of 600 articles published in three leading medical journals between 1946 (when the first random clinical trial was performed) and 1976 revealed a clear deterioration in study quality. In 1946, only 24 percent of the studies used existing data instead of collecting data specifically for the study; by 1976, the percentage was 56 percent. Longitudinal cohort studies (where patients’ progress is followed over a period of time) decreased from 59 to 34 percent. And sample sizes in most of the studies were considered too small, making the chance of detecting a difference in treatment highly unlikely. (Fletcher and Fletcher 1979) We often make similar mistakes, using data from other studies instead of generating our own. For instance, Barbara Kitchenham has repeatedly cautioned others not to combine her data sets, but software engineers continue to disregard her warning.

Objectivity in medical studies is sometimes questionable, too. For example, Chalmers (1982) studied 1157 papers on adult-onsite diabetes, written by 55 authors between 1963 and 1978. Nearly ten percent of the authors reporting results in favor of a particular drug acknowledged support by the drug’s manufacturers, whereas only two percent of the authors of contrary studies had such support. And he found a statistically significant association between criticisms of the drug trials’ negative results published in journals and the drug companies’ advertisements in those journals. This behavior occurs frequently in the software engineering realm. How many times does a researcher develop a reading technique, complexity measurement, or maturity model and then report on the results of empirical evaluations of it? And how many times do organizations jump to adopt a new technology without objective evidence of its effectiveness? We cannot continue to rely on subjective evidence like this when performing our risk assessments.

Even when we do careful, objective, well-documented studies, the issue of scale is often given little attention. We can see the importance of scale in the medical realm, where the non-linearity of dose response (that is, what works in the small does not always work the same way in the large) leads to uncertainty about how risks for lower doses can be extrapolated to risks for higher ones. Ellen Silbergeld (1986) points out, for example, that there are different uncertainties associated with using population risks as opposed to individual risks in decision-making. In the same way, we should always ask ourselves how risks observed or assessed on small projects scale up to medium or large projects.

6. Separate facts from values when you can

A final problem with conventional risk assessment is that it can never be value-free; the way we view the world colors our interpretation of facts. "The conviction that risk assessment can never be a value-free exercise led an NRC [US Nuclear Regulatory Commission] committee to recommend in 1983 that the functions of risk assessment and risk management should not be institutionally separated in the regulatory process, even though agencies should seek as far as possible to prevent risk-management considerations from influencing their risk assessments." (Jasanoff 1991) But a glance at any software risk assessment guidelines reveals the kind of partitioning shown in Figure 1, where risk assessment and risk management are indeed separated.

Ellen Silbergeld (1991) argues eloquently that separating these two risk steps leads to a false sense of objectivity. She explains that it is difficult to quantify risk if you are not aware of the possible consequences, and she points out that we tend to quantify in ways that reflect our values. For example,

> "Choosing the method of determining variance (and thus the range of estimates of risk) entails a decision concerning the relative value of over- and underestimating the true variance using the limited data at hand. Choosing the method of the 95 percent confidence limit is a decision that an error in calculating variance is not as serious in risk regulation as is actually overexposing human populations. Selecting the alternative, the maximum likelihood estimate or MLE method, places greater value on the experimental data themselves.
>
> Interestingly, although this method is frequently advocated by those who decry risk assessment as marred by unresolvable uncertainties, relying on MLEs to describe variance places more weight on both the reliability of the data and the accuracy of the methods to predict human response, because it inherently assumes a greater precision of the variance estimates." (p. 107)

Values and past history affect the perception of risk, too, and initial beliefs structure the way subsequent evidence is interpreted. For example, McGrady (1982) looked at how family practice physicians reacted to the reported results of clinical trials. He found that only one-third of the time did the physicians choose the best scientifically-validated treatment as published in the literature. There are many reasons for this, including a lack of time to read the studies carefully, a tendency for the authors to write their results in an obscure way, and the tendency of physicians to prefer their old methods of treatment. Software engineers also have limits on their time, and research papers tend not to be very accessible to practitioners. So even if researchers produce new, effective techniques and tools, practitioners will tend to rely on older, more familiar (albeit less effective) techniques. Our risk analyses must reflect these tendencies and deal with them, rather than assuming that the new technologies will be embraced immediately.

In addition, we must try to express risks in ways that fit the values and experiences of those who are making decisions based on them. For instance, a risk can be quantified in one of two seemingly-equivalent ways: as the risk of one death per million in the US population, or as the risk that a town of 25,000 people will be killed. To most readers, the first risk seems small but the second large; the only real difference is in the concentration of impact. Similarly, the same risk can be expressed in terms of mortality or of survival to evoke different responses. Thus, subtle changes in risk expression can have major impact on perception and decision-making.

Furthermore, people who are comfortable with their current values or beliefs are reluctant to change their opinions when faced with new knowledge. For example, when told that the risk of future oil spills is small, people may ignore the evidence, responding instead to their beliefs that the petrochemical industry has failed to act responsibly in the past. In the same way, people respond to notions of future software risk based on their experiences with software in the past. Gary Klein (1998) points out that people use metaphors and mental simulations to understand concepts and make decisions. For this reason, it is important for us to find appropriate metaphors to understand and express risk. For example, each of us can understand how long 3.5 meters is because we can compare it with the length of other things we

have experienced. But we have more difficulty understanding a lifetime risk of 0.035, because we have little context for understanding what the numbers mean. We must personalize the quantified risk to make it meaningful.

Next steps

There is risk in not paying attention to what others have learned from managing risk in other disciplines. As scientists, we like to think that we can make objective, accurate assessments of our projects' risks, and then deal with them in a fair and effective way. But in reality, there is much fuzziness and uncertainty associated with the risks themselves and with our understanding of how to address them. So what can we do to give our clients and ourselves more confidence in our risk assessments and plans?

First, we must make our estimation and risk analysis techniques more practical. By looking to the social, medical and environmental sciences, and to public policymakers and regulators, we can organize our risk management activities so that they take into account the values, beliefs, and biases of those affected by our software products and processes. There is a significant literature on risk presentation that can help us understand how to describe risks in ways that people can compare and contrast their options for dealing with them. And we can tie the risk presentation and decision-making to the more global business and technical decisions we make when building and maintaining software. That is, instead of a separate activity, risk management can be integrated into the technical and management activities we already perform when developing software.

More specifically, we can look at risk throughout the software life-cycle. What are the risks associated with each requirement? With each design choice? With each technology or tool? What are the risks involved when changing the design? Correcting a defect? Porting the system to a new platform? And what are the risks and uncertainties inherent in our effort and schedule estimation? Hazard analyses? Reliability engineering?

Secondly, we can examine more carefully the way in which we gather and evaluate evidence. What is the credibility of the supplier? How solid is the study? Does the evidence support the conclusions? How broadly can the results be extrapolated? And how does each piece of evidence contribute to the conclusions drawn from the whole body of evidence?

Finally, we need more flexible approaches to risk management that allow us to take advantage of new techniques and new information. Risk assessment, like effort and schedule estimation, is often performed at a project's launch and then never again during the project's lifetime. By using risk only to justify a project, we miss the wonderful opportunity of learning from our mistakes, and changing our plans as our understanding grows. We should revisit our risk assessment and revise our risk management plan whenever we learn something new and important about our products and processes. Our project post-mortem can also include a review of our risk assessment and management, to see where we could have anticipated a problem and handled it earlier, and to therefore update our checklists and review sheets so that we do not make the same mistake on subsequent projects. Insanity is sometimes defined as doing the same thing twice and expecting different results the second time. Good risk management can help ensure that we have sane software engineering practices.

Acknowledgments

I am grateful to Linda Greer of the Natural Resources Defense Council for her assistance in examining environmental risk management. A summary of problems in environmental risk management can be found in Rachel's Environment and Health Weekly, #420, Environmental Research Foundation, PO Box 5036, Annapolis, MD 21403.

References

1. Barry W. Boehm, "Software risk management: principles and practices, IEEE Software 8(1), pp. 32-41, January 1991.
2. T. C. Chalmers, "Informed consent, clinical research, and the practice of medicine," Transactions of the American Clinical and Climatological Association, vol. 94, 1982, pp. 204-212.
3. Commission of the European Communities, Benchmark Exercise on Major Hazard Analysis, 3 volumes, Commission of the European Communities, Luxembourg, 1991.
4. Lawrence Fisher, "Pentium flaw creates confusion for PC buyers," New York Times, December 14, 1994, pp. D1, D18.
5. R. H. Fletcher and S. W. Fletcher, "Clinical research in general medical journals: a 30-year perspective," New England Journal of Medicine, vol. 301, 1979, pp. 180-183.
6. IBM, "IBM halts shipments of Pentium-based personal computers based on company research," press release, December 12, 1994.
7. Intel, "Floating point flaw in the Pentium processor," memorandum, December 1, 1994, available from Intel's Faxback service, 1-800-628-2283 or 916-356-3105, document 7999.
8. Sheila Jasanoff, The Fifth Branch: Science Advisors as Policymakers, Harvard University Press, Cambridge, Massachusetts, 1990.
9. Sheila Jasanoff, "Acceptable evidence in a pluralistic society," in Deborah Mayo and Rachelle Hollander, Eds., Acceptable Evidence: Science and Values in Risk Management, Oxford University Press, 1991.
10. Gary Klein, Sources of Power: How People Make Decisions, MIT Press, Cambridge, Massachusetts, 1998.
11. Peter H. Lewis, "IBM halts sales of its computers with flawed chip," New York Times, December 13, 1994, pp. A1, D6.
12. John Markoff, "Flaw undermines accuracy of Pentium chips," New York Times, November 24, 1994, p. D1.
13. Deborah Mayo and Rachelle Hollander, Eds., Acceptable Evidence: Science and Values in Risk Management, Oxford University Press, 1991.
14. .G. A. McGrady, "The controlled clinical trial and decision-making in family practice," Journal of Family Practice, vol. 14, 1982, pp. 739-44.
15. Shari Lawrence Pfleeger, Software Engineering: Theory and Practice, Prentice Hall, 1998.
16. Paul Rook, Risk Management Tutorial, 1993.
17. William Ruckelshaus, "Risk in a free society," Risk Analysis, vol. 4, p. 161, 1984.
18. Ellen Silbergeld, "The uses and abuses of scientific uncertainty in risk assessment," Natural Resources and Environment, 2(17), pp. 57-59, 1986.
19. Ellen Silbergeld, "Risk assessment and risk management: an uneasy divorce," in Deborah Mayo and Rachelle
20. Hollander, Eds., Acceptable Evidence: Science and Values in Risk Management, Oxford University Press, 1991.
21. P. Slovic, B. Fischhoff and S. Lichtenstein, "Characterizing perceived risk," in Perilous Progress: Managing the Hazards of Technology, edited by R. W. Kates, C. Hohenemser and J. X. Kasperson, Westview Press, Boulder, Colorado, 1985, pp. 91-125.
22. S. Whittemore, "Facts and values in risk analysis for environmental toxicants," Risk Analysis, vol. 3, 1983, pp. 23-33.