②

Technical Report
CMU/SEI-92-TR-30
ESC-TR-92-030

Carnegie Mellon University
Software Engineering Institute

Software Development Risk: Opportunity, Not Problem

Roger L. Van Scoy

September 1992

[illegible]

Technical Report
CMU/SEI-92-TR-30
ESC-TR-92-030
September 1992

Software Development Risk: Opportunity, Not Problem

Roger L. Van Scoy

Software Risk Management Program

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESC/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Software Development Risk: Opportunity, Not Problem

Abstract

What is risk? What is risk management? What does risk management have to do with software? Noted software expert Tom Gilb says:

If you don't actively attack the risks, they will actively attack you [Gilb88, p. 72].

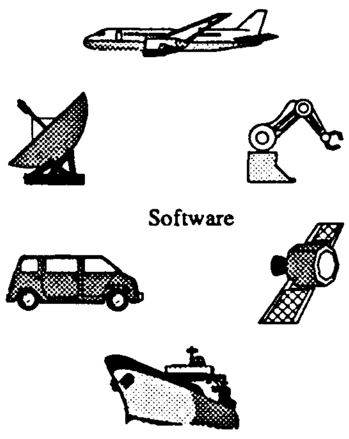
But what does it mean to actively attack risks? We answer these questions by examining the problems that exist in software development today and presenting the SEI Risk Program approach to turning risk into opportunity.

We begin by reviewing the fundamental concepts of risk and elaborating on how these basic concepts apply to the development of large, software-intensive systems. We then develop our strategy for seeing a systematic approach to risk management in software development be routinely practiced.

There are two key activities we are using to implement our strategy. The first is our risk management paradigm. The paradigm defines a set of continuous activities that must be undertaken to resolve technical risk in a systematic and structured way. The second is our risk assessment process for collaborating with clients to identify their technical risks.

We end with our ultimate goal: establishing an effective risk management ethic as standard practice in the software engineering industry.

Background

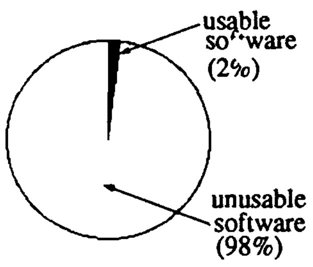


Why is software important? Software flies our airplanes [Tomayko91], controls our automated teller machines [Kanter90], and even controls our car engines [Woolnough90]. While software is prevalent in our everyday lives, it is the lifeblood of complex Department of Defense (DoD) systems. Consider the following:

The *Report of the Secretary of Defense to the Congress on the FY 1985 Budget* identifies 160 key programs....Of the programs identified in the report, at least 120 (or 75%) were determined to have a significant software component [Redwine84, p. 2].

Also, Robert Charette [Charette88] found that:

The AEGIS cruiser is a prime example of a system that could not operate without computers [and hence software]. It requires computers to help run every one of its major systems... [p. 12].



As software has taken a central role in DoD systems, the failures of software development efforts have come to the forefront. For instance, in a small sample of DoD programs (nine programs totalling $6.8 million) the Comptroller General [CompGen79] found that less than 2% of the software was usable as delivered. In addition, Charette [Charette88] chronicles a long list of cost overruns and failures in software-intensive programs.

Yet programs are planned to succeed. They are planned to produce the product needed by the customer, within cost and schedule. But there are many obstacles to their success. One key obstacle is the inability to see cost and schedule problems as symptoms of a more fundamental, underlying problem.

This underlying problem is often unresolved technical risk. It occurs because programs are unable to cope with technical risk in the development process. The introduction to a 1986 General Accounting Office (GAO) report entitled *Technical Risk Assessment* [GAO86] begins:

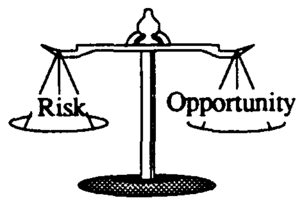
Technical risks are inherent in the development of new weapon systems, whose advanced performance requirements may exceed the capabilities of current technology. Not to anticipate technical risks before and during the development process creates the potential for scheduling and cost problems and, worse, the possibility that a system will fail to meet its design specifications and will not function as intended. In line with this, a 1983 Air Force report on an 'affordable acquisition approach' found technical problems a factor in more than 50 percent of the programs that experienced cost growth [p. 10].

Technical risks that become technical problems cause programs to fail. Since software has become critical in DoD systems, the effects of failing to manage software technical risks during development are magnified.

But what is technical risk? For that matter, what is risk? Everyone has an intuitive understanding of risk; it surrounds us in the world. But how can understanding risk help programs be more successful? First, we need to understand that a risk is not a problem. Rather, a risk is something that might occur in the future: a possibility, not a certainty. To be technically precise, there are two factors that comprise a risk:

1. Probability or likelihood that it will occur.
2. Loss resulting from its occurrence.

Risk is a part of any activity and can never be eliminated, nor can all risks ever be known. Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity.



So what is technical risk? Technical risk is the possibility that the application of software engineering theory, principles, and techniques will fail to yield the right software product. Technical risk is comprised of the underlying technological factors that may cause the final product to be:

overly expensive,
delivered late, or
unacceptable to the customer.

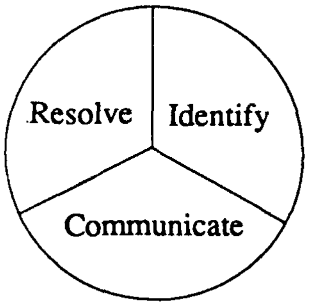
Technical risk lies at the heart of many problems causing the failure of software programs today. The essence of technical risk is the failure to build the right product. As the GAO [GAO86] pointed out, the ultimate consequence of a risk is that the delivered system will not perform as needed. The final risk always belongs to the customer.

As bleak as the picture is for system technical risk (as reported in [GAO86]), the picture for software is worse. There is very little experience in dealing with software technical risk. It is not clear how to identify, measure, or manage software technical risk. In the final analysis, little is known about software technical risk in the context of the overall system.

Where do these risks come from? Why aren't software risks identified and resolved before they become problems? Detecting software risks during development is especially difficult because we operate at the state of the art in our system developments and must continually cope with the uncertainty inherent in that environment.

So what can be done? According to Henry Petroski, "experience has proven that technology risks are controllable" [Petroski82]. What does that mean for software and the technical risks of developing software intensive systems? It should mean that if other engineering disciplines can control their technical risks, then software engineering is capable of doing the same.

What does it take to control software technical risk? We believe there are three key elements:



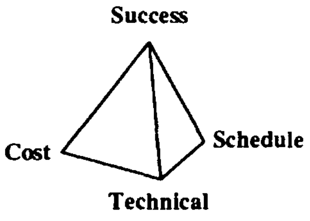
1. **Identify.** We must find the risks while there is still time to take action. This involves looking into the future and considering the path we have chosen from a risk perspective.
2. **Communicate.** We must accept that the risks exist and communicate the risks to those empowered to resolve them.
3. **Resolve.** We must make a conscious decision to act on the risks. This is turning risk into an opportunity to enhance our chances of success.

We call these three elements "risk management." The heart of risk management is informed decisionmaking under uncertainty. Risk management is about being active, not passive. Or as Charette [Charette91] has said, "Risk engineering [management] does not deal with future decisions, but with the future of present decisions."

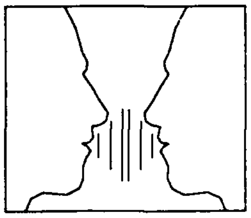
Most software development managers see their job as managing risk. Unfortunately, these managers are seldom given the information they need to effectively manage their program's risks. The result is reactive crisis management based solely on cost and schedule indicators. Few managers are systematically identifying, analyzing, planning, tracking, and controlling their technical risks. When managers address technical risks, the issues tend to be based solely on their own experience, and their methods are generally incomplete and undocumented.[1] For software risk management to be effective, we must get away from the ad hoc, "experience is the only teacher" approach that currently dominates software risk management.

1. These findings are based on the field work of the SEI Risk Program and are discussed in detail in [Kirkpatrick92].

Yet we should not fool ourselves into thinking that risk management is the next "silver bullet."[2] To be successful, a program manager must be able to balance all the program concerns, among which are cost, schedule, and technical considerations. Managing cost and schedule are just as important as managing technical risk.[3] To be effective, risk management must be an integral part of the way programs are managed by the contractor and the customer, not just in terms of cost and schedule risk, but in terms of technical risk as well. If risk management is going to help programs succeed, then there needs to be a means to assess and manage the technical risk in programs.



Unfortunately, the difficulties of applying risk management do not end with its incorporation into program management. Even when the risks are known, we have found strong cultural barriers that prevent risk communication.[4] This problem exists within programs, where everyone belongs to the same corporate team, as well as between the contractor and the customer, where the current environment leads to unshared and competing program objectives. If risk management is to be effective, the culture that inhibits people from admitting that there might be a risk (or a problem) must be changed.



The good news is that the evidence from other disciplines suggests that managing technical risk during software development can result in more successful software programs. The strategy the SEI Risk Program has developed to make this happen is the topic of the next section.

Risk Program Strategy

Our strategy can be summarized by the following statement:

> Until we use a disciplined and systematic way to identify and confront technical risk, we will never be able to control the quality, cost, or schedule of our software products.

This does not mean that programs that practice risk management will not have problems or even fail. But risk management helps programs suc-

2. See [Brooks87].
3. Again from [GAO86], "GAO believes that the findings [the GAO's findings] demonstrate a need for more clarity in, and attention to, technical risk assessment in the DoD. The findings do not suggest that technical risk is more critical than cost or schedule risk, or that DoD's attention to cost or schedule risk be reduced" [p. 5].
4. See [Kirkpatrick92] for more details.

ceed by providing them with the tools to understand and make better decisions about the technical risks that may prevent their success.

The goal of the Software Risk Management Program at the SEI is to help software programs succeed. We believe one way to achieve this goal is to improve the practice of risk management for software-intensive systems so that customers and suppliers routinely use a structured process for managing and communicating program risks throughout the entire program life cycle.

We have chosen to focus on technical risk in the software development process because very little is understood about software technical risk. We have chosen to focus on software because it is such a key part of many DoD systems. However, to properly address software technical risk requires a system perspective, because software never exists in a vacuum. Recognizing this, we have chosen to focus on software technical risk in the context of the total system.

Our strategy consists of the four phases shown in Figure 1.

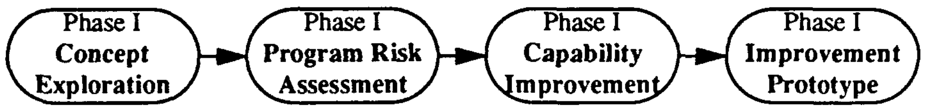


Figure 1. Risk Program Phases

In Phase I we determined that software risk management would be an effective area for the SEI to address and a valuable asset to our clients.

In Phase II, we have developed a risk management paradigm and risk assessment process (discussed in the next two sections). These activities include: building an awareness of software risk management within the software development and risk management communities; developing mechanisms and methods for applying our risk management paradigm; testing these with our strategic partners[5] by risk assessments; and refining our paradigm, mechanisms, and methods according to how well they work in practice.

5. A strategic partner is an organization that works collaboratively with us to test and refine our approach to software risk management and has a long-term commitment to adopting risk management practices.

In Phase III, we will continue to refine our risk management paradigm, its mechanisms, and methods. The emphasis in Phase III shifts from testing by risk assessments to pilot capability improvement[6] efforts.

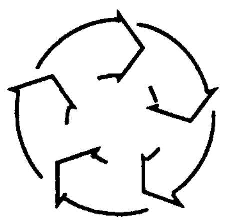
Finally, in Phase IV, we will move from working with individual programs to working with whole organizations. Here, we want our risk management paradigm, methods, and tools to be institutionalized in the program management process of our strategic partners.

By the end of Phase IV, we will have developed and tested a balanced approach, using both qualitative and quantitative methods, to software risk management. Our strategy is based on an evolutionary approach that builds credibility and awareness through interaction with actual programs and hands-on application. We are dependent on customer contact throughout this work; strategic partnerships establish our operating base and apply their experience and knowledge to software risk management. As we develop the foundation for software technical risk management, we will work with our clients as they seek to apply risk management in their programs.

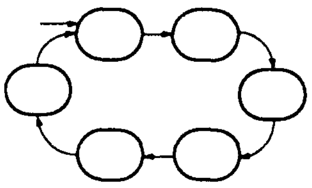
To achieve our goal of helping programs succeed through the application of risk management, we have established two objectives that meet the needs described previously. These objectives are:

1. To establish a foundation for addressing software-related risk.
2. To strengthen the ability of organizations to evaluate and manage software-related risk.

The key to implementing software risk management is to establish the needed foundation. Our first step toward that foundation is the creation of a risk management paradigm, which is our disciplined and systematic approach for managing software risk. As this paradigm is implemented with specific risk management methods, we are testing the paradigm to understand what will or will not work in the real world. As we test, we collect information on the best practice in software risk management and the kinds of risks programs encounter. We are analyzing this information base for common risks that span industry sectors and program types.


Risk Management Paradigm

6. Capability improvement is incorporating our risk management paradigm into program management on pilot programs.


Risk Assessment Process

The ability of organizations to evaluate and manage software-related risk requires a baseline from which to work. To create this baseline, we have developed a risk assessment process that organizations can use to identify and start managing their software development risks. From this baseline, we plan to pilot efforts to integrate software risk management into program management and software process management on individual programs. These pilot integration efforts will lead naturally into risk management improvement at the organizational level.

Throughout this entire effort we are interacting with the larger software community for input and feedback. We are interviewing program staff from industry and government to determine what people know about software risk management, how they approach software risk management, and how software risk management can fit into program management. These interviews help us formulate our vision of the future. As a broader forum, we are conducting conferences and workshops to build awareness of software risk management.

We can summarize the steps in our strategy as:

1. Define software risk management by research and interviews.
2. Develop an effective software risk management paradigm by testing and prototyping with individual programs.
3. Transfer software risk management from successful programs to organizations and the larger community.

To better understand how our strategy is being implemented, we next discuss our risk management paradigm and risk assessment process in more detail.

Risk Management Paradigm

The risk management paradigm, shown in Figure 2, is our view of a repeatable process for software risk management. The paradigm is a model of how the different elements of software risk management interact and a framework for describing how software risk management can be implemented. The paradigm has a circular form to highlight its continuous nature. The arrows signify the logical flow of information between the elements of the paradigm. Communication is the core. It is the means by which all the information flows.

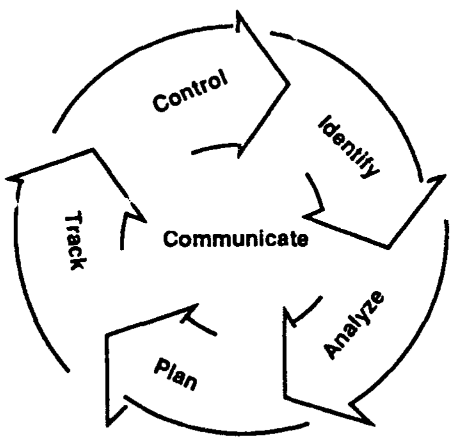


Figure 2. Risk Management Paradigm

We can summarize the elements of our paradigm as:

Identify – locate risks before they become problems and adversely affect the program.

Analyze – turn the raw risk data into decisionmaking information.

Plan – turn the risk information into decisions and actions (both present and future).

Track – monitor the status of risks and actions taken against risks.

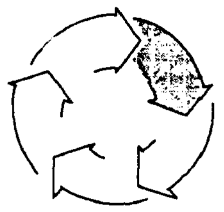
Control – correct for deviations from the planned risk actions.

Communicate – provide the feedback on the active risk activities, current risks, and emerging risks among the paradigm elements and within the program.

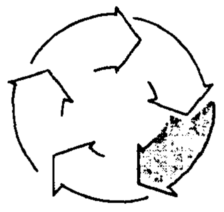
For each element in the paradigm, we plan to provide guidance on specific implementation techniques. These techniques will come from existing practice or will be developed specifically to support the risk paradigm.[7] In essence, the paradigm is a framework that encapsulates the best practice in software risk management. Using this framework, a program will draw together the pieces of risk management practice that fit

7. Much is known about risk management in other domains, such as medicine, nuclear energy, and hardware reliability. There is also a large body of knowledge in decision theory that deals with decisionmaking under uncertainty. Researchers in the social and psychological sciences have done extensive work on how to elicit sound data from people. These domains have useful insights that are needed for constructing different elements of the paradigm. We plan to adapt the proven techniques from these (and other) domains and apply them to the management of software risk.

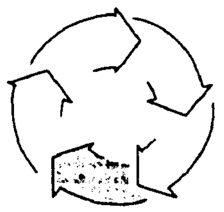
into its program management structure. A brief overview of each paradigm element is presented below.



Risk identification is the first element in the risk management paradigm. Before risks can be managed, they must be identified. Risk identification aims to find the major risks before they adversely affect a program. We are developing techniques to discover risks by exploiting and communicating the risk knowledge of the program team.



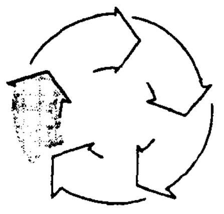
Risk analysis is the next element in the risk management paradigm. Risk analysis is the conversion of risk data into risk management information. Each risk must be understood sufficiently to allow a manager to make decisions. Risk analysis sifts the known risks, and places the information in the hands of the decision maker. Analysis provides the information that allows managers to work on the *right risks*.



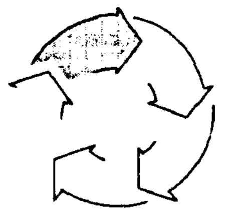
Risk planning is needed after a risk is identified and analyzed. This element includes developing actions to address individual risks, prioritizing risk actions, and orchestrating the total risk management plan. An individual risk action plan could take many forms, for example:

- Mitigate the impact of the risk by developing a contingency plan (with a triggering event) should the risk occur.
- Avoid a risk by changing the product design.
- Accept the risk and take no further action, thus accepting the consequence if the risk occurs.
- Study the risk further to acquire more information and better determine the uncertainty or loss associated with the risk.

The key to risk action planning is to consider the future consequences of each decision made today.

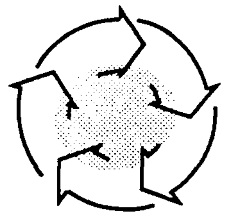


Risk tracking is required to ensure effective action plan implementation. This means that we must devise the risk metrics and triggering events needed to ensure that the planned risk actions are working. Tracking is the watch dog function of the risk action plan.



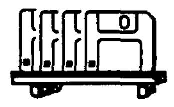
Risk control is the next element in our paradigm. Once the risk metrics and the triggering events have been chosen, there is nothing unique about risk management. Rather, risk management melds into program management and relies on program management processes to control the risk action plans, correct for variations from the plans, respond to triggering events, and improve the risk management process. In fact, if risk management is not integrated with day-to-day program m^nagement, it will soon be relegated to an ineffectve background activity.

Finally, **risk communication** is at the center of our paradigm because, without effective communication, no risk management approach is viable. Communication is critical because it facilitates interaction among the elements of the paradigm. But there are higher level communications to consider as well. Risks must be communicated to the appropriate organizational levels so the risks can be analyzed and managed effectively. This includes levels within the development organization, within the customer organization, and most especially, across that threshold between the developer and the customer.



Example

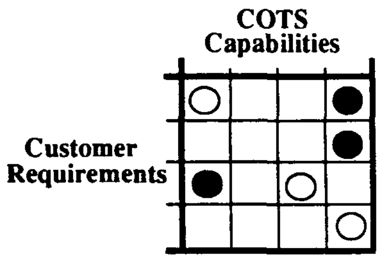
As an example of applying the risk management paradigm, consider the use of commercial off-the-shelf (COTS) software in a system. COTS software is often viewed as a means to manage system risk by reducing the amount of new software developed on a program or by incorporating a working design element into a system.



But does the use of COTS software really reduce system risk? Conventional wisdom would say yes, but the answer may not be that obvious. How could we identify the risks associated with using COTS software?

We could take each COTS product and map the capability of the product to the customer requirements we are trying to meet. For each of those intersections, we might consider the following questions:

- Can software developers use the COTS product to implement this requirement?
- What if the COTS product does not meet the requirement?
- What if the requirement changes: will the COTS product still provide the required functionality?
- What if the vendor changes the COTS product: will it still be compatible with the rest of the system?
- What if the COTS vendor does not deliver on time?



Next, we need to analyze the situation. Suppose the vendor has a reputation for changing products. Could this be a problem down the road? How big of a problem might this be?

If the analysis shows that changes to this particular COTS product could be bad for the program, then we need a plan to deal wi᷊ this risk. Is there some action we can take today to remove the risk? Is there some action we should take later if it appears that the product will change? How does handling this particular risk fit into the program plan?

As the COTS example illustrates, risk management requires a long-term systems perspective. Risk management is not a one-time activity. It is an ongoing activity that must be an integral part of the way we do business. To obtain the full benefit of risk management, it must be a continuous process that is integrated into the way we manage programs.

Risk Assessment Process

Risk assessment provides a snapshot of the risk situation and is part of a viable risk management program. Although snapshots are valuable, effective risk management requires continuous vigilance and application–a risk ethic. A risk ethic involves everyone and is a continuous process of identifying, communicating, and resolving risks in an open and non-threatening environment.

For the Risk Program, the risk assessment process, shown in Figure 3, is the first application of our risk management paradigm.

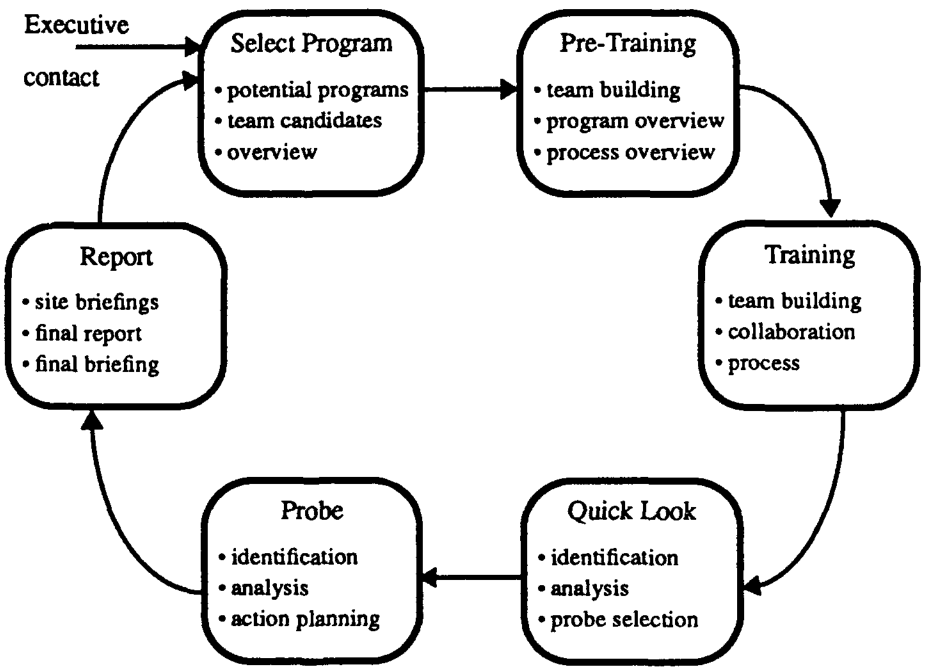
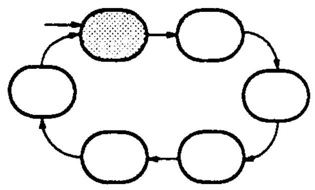
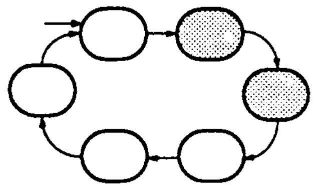


Figure 3. Risk Assessment Process

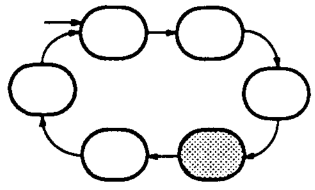
To execute a risk assessment, we have drawn selected risk management elements from the paradigm and put them together in a way that allows a team of trained software professionals to assist a program in identifying and managing risk.



The risk assessment process begins with executive management buy-in and cooperation. Only executive management has the authority and resources to make risk management a success.[8] In a risk assessment, the risk assessment team has both client organization and SEI Risk Program members. The team then works collaboratively with a single program within the client organization. This gives the team a technical, product-oriented focus and allows it to work in detail with people at all levels of the program, both customer and contractor. The role of the assessment team is to facilitate, that is, the team's job is to help the client program work through the process of identifying and managing program risks.



The risk assessment begins with team training. The team meets prior to the risk assessment for team building and training in the details of our risk management paradigm and risk assessment process. This training includes instruction in the risk management mechanisms to be applied during the assessment as well as practice exercises using the mechanisms. Another important part of the training period is familiarizing the assessment team with the program to be assessed. This takes place in two ways: by the interaction among the team members and by program briefings given by the client organization team members.



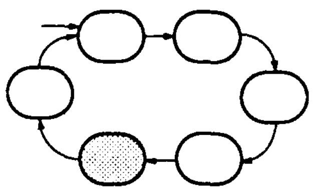
The first on-site activity is referred to as Quick Look. The Quick Look is designed to be just that–a quick, top-level look at the entire program.We examine the software from a system perspective, giving the assessment team a general, system-level understanding of the program. The assessment team seeks to identify broad areas of risk for the next on-site period (referred to as Probe). During Quick Look, the assessment team:

- Applies selected risk identification and analysis mechanisms.
- Consolidates the results across the different mechanisms.
- Identifies possible risk areas for further investigation.
- Briefs program participants on the Quick Look findings.

The Quick Look ends with a briefing that outlines areas that the assessment team feels need further investigation. The final decision on when to

8. For additional verification, see [Humphrey89].

proceed with the Probe and which areas to investigate belongs to the program manager.



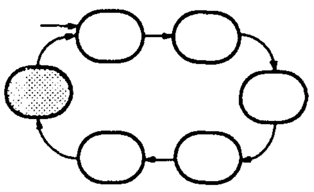
The Probe on-site period is an in-depth investigation into the risk areas selected by the program manager. It is here that the actual risks a program may have in the probe areas and what the program may do about them are identified. During Probe, the assessment team:

- Applies selected risk identification mechanism(s) in depth to the selected probe areas.
- Analyzes the identified risks.
- Identifies possible risk actions that could be taken.
- Briefs the program participants on the Probe findings.

Again, the results of the Probe belong to the program manager. Table 1 compares the Quick Look and Probe activities.

Table 1: Quick Look and Frobe Comparison

	Quick Look	Probe
Input	Participants Request for Proposal Requirements documents System design documents	Participants Detailed design documents Requirements documents System design documents
Participants	System engineers Software engineers Functional area specialists Program management	System engineers Software engineers Functional area specialists Program management
Activities	Identification Analysis (limited)	Identification Analysis Action planning
Output	Risk areas Possible sources of risk	Specific risks Possible risk actions



Once the Probe on-site period is complete, the risk assessment concludes by bringing the results of Quick Look and Probe together into one package for the program manager. The results of the risk assessment belong to the sponsoring program manager. No assessment results are attributed to any group or individual within the client program. While this reporting step is the conclusion of the risk assessment process, it is just the beginning of our strategic follow-up process. Our strategy is to continue work with assessment clients to evaluate the long-term impact of risk management and to assist our clients in adopting risk management as a regular, routine activity.

Conclusion

Risk does not have to be negative. In fact, knowing our risks provides opportunities to manage and improve our chances of success. However, this view of risk is only possible in a "risk aware" culture where risks can be communicated freely and openly. Indeed, changing the way people work and think is no small feat. It can only be accomplished by demonstrating that software development risk management works, and demonstrating it in a way that is not threatening or burdensome to managers or developers.

Fortunately, the evidence from other engineering disciplines strengthens our belief that software technical risks can be managed. Risk management must become a routine part of the software development process. Risk management cannot be an audit, a check mark on a standard, or something done only during "risk management season."

The key to reducing the surprises in our software development efforts is to incorporate a disciplined and systematic approach to managing technical risk into the software development process. We cannot escape risk, but risk management can equip us to deal more effectively with the future, because,

> "The only thing harder than predicting the future is changing the past." [9]

Acknowledgments

This paper is a summary of the hard work by all the members of the SEI Software Risk Management Program.

9. Source unknown.

References

[Brooks87]	Brooks, F.P. "No Silver Bullet: Essence and Accidents of Software Engineering." *IEEE Computer, 20*, 4 (April 1987), 10-19.
[Charette88]	Charette, R.N. *Software Engineering Risk Analysis and Management*. New York: McGraw-Hill, 1988.
[Charette91]	Charette, R.N. *Information Technology Risk Engineering*. SEI/NSIA Workshop on Software Risk, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, February 1991.
[CompGen79]	Comptroller General. *Contracting for Computer Software Development*. FGMSD-80-4, Government Accounting Office, Washington, D.C., September 1979.
[GAO86]	Government Accounting Office. *Technical Risk Assessment: The Current Status of DOD Efforts*. GAO/PEMD-86-5, Government Accounting Office, Washington, D.C., 1986.
[Gilb88]	Gilb, T. *Principles of Software Engineering Management*. Addison-Wesley, 1988.
[Humphrey89]	Humphrey, W.S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.
[Kanter90]	Kanter, J., S. Schiffman, and J.F. Horn. "Let the customer do it; from grocery robots to photo kiosks, computerized self-service is in." *Computerworld, 24*, 35 (August 1990).
[Kirkpatrick92]	Kirkpatrick, R.J., J.A. Walker, and R. Firth. "Software Development Risk Management: An SEI Appraisal." *1992 SEI Technical Review*, R.L. Van Scoy, ed. Software Engineering Institute: Carnegie Mellon University, Pittsburgh, PA, 1992.
[Petroski82]	Petroski, H. *To Engineer is Human: The Role of Failure in Successful Design*. New York: St. Martin's Press, 1982.
[Redwine84]	Redwine, S.T., Jr., L.G. Becker, A.B. Marmor-Squires, R. J. Martin, S.H. Nash, and W.E. Riddle. *DoD Related Software Technology Requirements, Practices, and Prospects for the Future*. IDA Paper P-1788, Institute for Defense Analyses, Alexandria, Virginia, June 1984.
[Tomayko91]	Tomayko, J.E. "The Airplane as Computer Peripheral." *American Heritage of Inventions & Technology, 7*, 3 (Winter 1991).
[Woolnough90]	Woolnough, R. "TI drives at Euro autos; makes 'PACT' to win microcontroller business." *Electronic Engineering Times* (August 1990).

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-30	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-92-030
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (City, State and ZIP Code) ESC/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO 63756E; PROJECT NO. N/A; TASK NO N/A; WORK UNIT NO. N/A

11. TITLE (Include Security Classification)
Software Development Risk: Opportunity, Not Problem

12. PERSONAL AUTHOR(S)
Roger L. Van Scoy

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM TO	September 1992	20

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)
Software Risk Management
Software Development
Risk

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

What is risk? What is risk management? What does risk management have to do with software? Noted software expert Tom Gilb says:

If you don't actively attack the risks, they will actively attack you [Gilb88, p. 72].

But what does it mean to actively attack risks? We answer these questions by examining the problems that exist in software development today and presenting the SEI Risk Program approach to turning risk into opportunity.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL Tom Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631 22c. OFFICE SYMBOL ESC/AVS (SEI)

ABSTRACT —continued from page one, block 19

We begin by reviewing the fundamental concepts of risk and elaborating on how these basic concepts apply to the development of large, software-intensive systems. We then develop our strategy for seeing a systematic approach to risk management in software development be routinely practiced.

There are two key activities we are using to implement our strategy. The first is our risk management paradigm. The paradigm defines a set of continuous activities that must be undertaken to resolve technical risk in a systematic and structured way. The second is our risk assessment process for collaborating with clients to identify their technical risks.

We end with our ultimate goal: establishing an effective risk management ethic as standard practice in the software engineering industry.