



Seven Characteristics of Dysfunctional Software Projects

Michael W. Evans, Alex M. Abela, and Thomas Beltz
Integrated Computer Engineering, Inc.

Taking advantage of its many years of experience in identifying and evaluating project risks in large-scale software systems acquisition and development programs, Integrated Computer Engineering has developed a risk database. Their analysis of this risk database has identified seven predominant characteristics that provide insight into the causes of dysfunctional software projects. This article identifies these characteristics and the typical real-world risks that accompany each.

Integrated Computer Engineering (ICE), Inc., a subsidiary of American Systems Corporation, is experienced in identifying and evaluating project risk in large-scale systems acquisition and development programs. During the last 12 years, ICE assessed more than 280 federal, state, Department of Defense, and commercial software-intensive programs and projects. These projects were responsible for the acquisition or development of leading-edge weapons, communications, financial, logistics, and public service automated data processing systems.

The project risks that were collected during ICE's 12 years of project assessments began to form a substantial database of useful information. Also added to this risk database are project risks gathered from risk studies conducted by the Institute for Defense Analysis [1], risks identified by Capers Jones [2] and Tom DeMarco [3], and from risks identified during risk management services ICE performed in support of the Software Program Managers Network [4]. To date, the ICE risk database has grown to more than 800 primary and secondary project risk indicators.

What emerged from the ICE risk database were seven predominant characteristics relating directly or indirectly to common failures observed among those system acquisition and development projects that had the greatest difficulty delivering a quality product on time and on budget. The seven common characteristics are listed below:

1. Failure to Apply Essential Project Management Practices. Many troubled projects fail to apply proven project management disciplines like cost estimation, project scheduling, resource planning, configuration management, and proactive risk management, then wonder why their project is in constant turmoil.
2. Unwarranted Optimism and Unrealistic Management Expectations. Some managers recognize the potential for

negative impact on their project from potential problem areas; however, they choose to see things through rose-colored glasses, assuming that problems will work themselves out even when all available evidence raises the red flag.

3. Failure to Implement Effective Software Processes. Many projects fail to implement effective software processes because their approach to process application is not balanced. Some apply minimal process and rely on staff expertise, while others insist on rigorous global process conformance.

“Managing a software project requires ‘courageous’ and often clairvoyant individuals ... willing to confront today’s challenges to avoid tomorrow’s catastrophes.”

4. Premature Victory Declarations. Pressures to deliver timely products often result in premature declarations of completion by managers. Success cannot be declared until products have been completed with the built-in contracted quality and reliability.
5. Lack of Program Management Leadership. Managing a software project requires “courageous” and often clairvoyant individuals who are willing to confront today's challenges to avoid tomorrow's catastrophes. We have observed two types of problem managers: those with software engineering and no management experience, and those with management and no software engineering experience. Both

types lack the ideal blend of both technical and managerial know-how.

6. Untimely Decision-Making. Some managers avoid making time-critical decisions until it is too late, even when they are faced with overwhelming warning signs of impending problems.
7. Lack of Proactive Risk Management. Many projects claim to implement risk management but few do so effectively. “What distinguishes the best organizations and best managers is not just how well they do in their successful efforts, but how well they contain their failures [5].”

Now, let's take a closer look at some real-world risks that have been associated with each of the seven characteristics.

Failure to Apply Essential Project Management Practices

Typical risks are as follows (risk designators are listed in Table 1, page 19):

- (A) The process being followed and decisions being made will result in a product that may not satisfy the critical needs of the user and are inconsistent with the severity of the consequences of project failure.
- (I) Project plans do not describe how technology will be used resulting in a need to continuously rework inconsistent products and correct resulting problems.
- (J) Software reliability problems will not be discovered because procedures are not established for the collection and analysis of error data generated during software development.
- (C) Project plans are unrealistic or not implemented and do not result in a predictable development environment.
- (K) Software defects will not be found because the contractor has neither conducted nor planned for software design inspections or walkthroughs.
- (L) Essential system functions do not perform adequately or reliably due to

testing problems or insufficient testing of key software components.

What we repeatedly find through assessments is that while the mainstream software tasks have been reasonably well planned and implemented, certain essential project management practices are not. The practices that are routinely at the bottom of list are: cost estimation, scheduling, resource planning, configuration management, risk management, earned value reporting, performance-based metrics, re-estimation, quality assurance, and rigorous testing.

Some managers perceive these practices as bureaucratic red tape that only gets in the way of real engineering. Also, methods such as risk management, metrics, and re-estimation often provide managers with more reality than they care to know or handle.

Unwarranted Optimism and Unrealistic Executive Management Expectations

Typical risks are as follows:

- (A) The process being followed and decisions being made will result in a product that may not satisfy the critical needs of the user and are inconsistent with the severity of the consequences of project failure.
- (B) The staff is not capable of implementing the product and applying the technologies selected. Excessive turnover may impact project success.
- (C) Project plans are unrealistic or not implemented and do not result in a predictable development environment.

In some projects there is an underlying belief that all will be well. The reality is that planning for the worst and being surprised when it does not occur is a much more effective way to manage a software project. In 1995, only 16 percent of software projects were expected to finish on time and on budget. An estimated 53 percent of projects cost nearly 190 percent of their original estimates [6]. When managing or participating in a system acquisition or development project there is absolutely no rationale for optimism. Historical data do not support an overly confident posture when managing large complex high-tech programs. Why then, is this unfettered optimism so common?

Two principal causes of unwarranted optimism have been observed. The first relates to the second degree of ignorance, or “not knowing what you don’t know.” Staff members with insufficient experi-

ence may have unrealistic optimism about success for the following reasons:

- They are not aware of the magnitude of the tasks or the problems they are attempting to solve.
- They oversimplify what it will take to achieve the required result or product.
- They attempt to implement silver-bullet technology solutions without having thoroughly evaluated their effectiveness or impact on the program.

The second cause of unwarranted optimism stems from unrealistic executive management expectations. “There is a major cultural barrier to accurate estimation [and scheduling] that must be highlighted ... If an early estimate [or schedule] predicts higher cost, longer schedules, or lower quality than client or manager expectations, there is a strong tendency to challenge the validity of the estimate. What often occurs in this situation is that the project manager is directed to recast the estimate so that it falls within preset and arbitrary boundary conditions [7].”

This self-imposed cultural barrier that some executive managers place between

*“The reality is that
planning for the worst
and being surprised
when it does not occur is
a much more effective
way to manage a
software project.”*

themselves and their program managers forces those managers to report unrealistic estimates, schedules, and project risks to their customers and to oversight organizations.

The cost of runaway or defective systems often gets personalized in the dismissal or demotion of the responsible executive.

I don’t want yes men around me.
Tell me what you think even if it
costs you your job.

– Louis B. Mayer, legendary head
of production at MGM

With the threat of removal hanging over their heads, many managers establish a “can-do-at-all-cost” mentality. Bad news is not tolerated, projections of

problems are not acceptable, and anything other than full steam ahead is punished in the severest manner.

Failure to Implement Effective Software Processes

Typical risks are as follows:

- (G) The technical process being used is inconsistent with the project’s requirements and the staff’s ability to implement it.
- (D) Design and code defects will not be discovered until late in the development – too late to avoid cost, schedule, or quality impacts.
- (H) The design (system or software depending on where the indicator is observed) may not support the application’s critical safety or security requirements.
- (F) There is inefficient software development due to failure to allocate requirements early in the design phase.

Many software projects’ managers assume that since trained software engineers staff the project, project-specific standards, guidelines, and common tools are unnecessary.

There are two factors at work here that impact the ability of the project to apply common processes to specific projects. The first is project uniqueness. To paraphrase Tim Lister, each project is unique [8]. Each has its own quirky clients, its own unique staff, and its own expectations of success. Could it be that adaptation of process is 90 percent of the problem and the common processes are marginal? Technology and process are not a “cookie-cutter” solution to every development problem; the key to success is adaptation of the technology and process to meet the unique challenges of a specific project or program.

“With the right people you might succeed without process maturity, but ... the best process in the world will not make you successful without the right people [9].”

The second factor is project balance. Technology, tools, processes, and people must all be in balance at the project, not the organizational level.

Premature Victory Declarations

Typical risks are as follows:

- (L) Essential system functions do not perform adequately or reliably due to testing problems or insufficient testing of key software components.
- (N) The system may not satisfy the needs or expectations of the user when delivered.

- (O) Early release of unqualified products results in unexpected failures, failures in key user areas, and potentially corrupted data, which destroys confidence in future releases.

We have observed that the pressure to deliver timely product has a tendency to overwhelm the need for quality and consequently results in an early and unwarranted victory declaration by management. "... Some of my staff would tinker on a task forever, all in the name of quality. But, in some cases the market doesn't care that much about quality. Instead, it's screaming for the product to be delivered yesterday and is willing to accept it even in a quick and dirty state. The decision to pressure people into delivering a product that doesn't measure up to their own quality standards is almost always a mistake [9]."

A clear understanding of the customer's quality expectations is an essential prerequisite to client satisfaction. Delivery of a faultless solution that is significantly over-budget and so late in delivery as to be obsolete will fail to satisfy a customer as much as a product delivered on time that does not meet specified requirements or that has poor reliability.

Lack of Program Management Leadership

Typical risks are as follows:

- (C) Project plans are unrealistic or not implemented and do not result in a predictable development environment.
- (E) The planning has not been updated recently and now is out of date with the project environment and does not reflect current agreements or constraints.
- (M) Customer relationships cause an environment that is unstructured and precludes successful implementation of a product within cost and schedule.
- (B) The staff is not capable of implementing the product and applying the technologies selected. Excessive turnover may impact project success.

"Poor project management will defeat good engineering, and is the most frequent cause of project failure [10]." Too many people who have never developed software are making decisions about how software should be developed. Attributes of a good software project manager include a broad range of technical software development experience, the ability to manage people and the dynamics of a team environment, and the willingness to proactively manage project risk and make timely decisions. To paraphrase Tom DeMarco, "Managers ... make the craziness go away [11]."

During a recent assessment of a severe-

ly troubled software project, the managers did not know the status of the product, the staff was demoralized, and the project was severely over budget and behind schedule. What we discovered will amaze you. Management had set a goal for this site to become Software Engineering Institute, Capability Maturity Model® (CMM®) Level 3 compliant by a certain deadline. Unbelievably, they diverted 40 percent of the experienced engineering staff to work the CMM issue. The manager said, "We thought the rest could take up the slack." This isn't rocket science. If you want to manage a software project you have to hunker down and do it right: no shortcuts, no nonsense, no silver bullets; just a laser beam to the finish line.

Untimely Decision-Making

Typical risks are as follows:

- (D) Design and code defects will not be discovered until late in the development – too late to avoid cost, schedule, or quality impacts.
- (E) The planning has not been updated recently and now is out of date with the project environment and does not reflect current agreements or constraints.
- (F) There is inefficient software development due to failure to allocate requirements early in the design phase. "Management is the art of planning work so that it can be accomplished within constraints of time, cost, and other resources at a level that will be competitive in the marketplace [12]." Delays caused by slow decision making erode these constraints even further while the project team waits for clear direction or crucial resources. Unless plans remain current, a project can be caught off guard when unexpected problems arise, leaving managers with insufficient controls, discipline, and/or support facilities to make effective and informed decisions.

A second problem results from late decision making. Simply stated, you can fix a bad decision, but no action occurs while projects wait for managers to decide what to do. During many project assessments we conducted, engineers have stated that they knew what actions to take and were ready to proceed, but could not move out until management decided the prudent course.

"Fast decision-makers often make better decisions than slow decision-makers," according to a study by Kathleen Eisenhardt of Stanford University and Jay Bourgeois of Virginia University. Fast decision-makers "set up systems to collect a range of information on their business and markets constantly, and then make deci-

sions using the data available. Slow decision-makers first analyze a problem and sort out the questions that must be answered. Only then, do they go out and look for that information [13]."

Lack of Proactive Risk Management

Typical risks are as follows:

- (P) Miscellaneous risks not being tracked make project success unlikely.
- (Q) Risk management may not prove effective or identify key risks.
- (R) The lack of an effective risk management process results in unplanned problems impacting the project.

"The problem of project management, like that of most management [is] to find an acceptable balance among time, cost, and performance [14]." When a project moves out of balance, a risk results. Often schedule performance becomes the most important issue due to customer pressures, resulting in a loss of focus on cost and product performance and increased project risk. "An effective risk management program is dynamic and ongoing throughout the development process and requires the participation of everyone involved [15]."

During our assessments, a significant amount of time is spent on determining the effectiveness and degree to which a project implements risk management as part of its management structure. In our experience, this assessment area is a bull's-eye indicator of the potential for overall project risk. Projects that fail to do an effective job of managing risk are constantly reacting to problems, while those that manage risk well anticipate rather than react. "Your organization will be much better once it moves away from reacting to change, and toward proactive anticipation and management of change [16]." To maximize potential for success, risk management should play a visible and key role in the process of project management.

"Risk management transcends modern management theory, such as Total Quality Management and Business Process Re-engineering, because it is basic to decision making. Risk management is based on theories that provide different strategies for decision making under problematic conditions [17]."

Analysis

Table 1 shows each of the seven risk characteristics and their respective risk designators (an upper case letter over the range A to R). Each risk event in the ICE database was characterized against the risk designators, and the tallied results are shown in the

third column from the left. The risk designator events were accumulated for each risk characteristic (fourth column), and the frequency of occurrence relative to all observed events in the database was calculated (far right column). It should be noted that the percentages in the far right column do not total 100 percent, as the risk designators are not unique to each characteristic.

The ranking of the characteristics is by frequency of characteristic occurrence; therefore, the data show what may be the likely dysfunctional causes, but not their relative impact on projects or programs.

Conclusion

When reviewing dysfunctional software projects, a reasonable approach would be to consider the risk descriptions for each of the seven characteristics we have identified and determine whether they apply.

Why do projects not address these issues if they are so apparent? The first reason is denial. When you are fighting the day-to-day realities of a software project, it is very easy to assume that the indicators of disaster are probably wrong, and the project will not be impacted the way the other 12 projects were. Denial is the excuse that enables program managers to make dumb decisions.

The second reason is cultural barriers. Coincidentally, all of the seven factors we identified focus on cultural, rather than technical, issues. "Since 1979 we have been contacting whoever is left on the project staff to find out what went wrong. For the overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to explain the failure [18]." Factors such as the seven we addressed here do matter, and they should be considered essential components of any project.◆

References

1. Technical Risk Indicators for Embedded Software Development. Institute for Defense Analysis, Paper P-3027, Oct. 1994.
2. Jones, Capers T. Assessment and Control of Software Risks. New Jersey: Yourdon Press, Feb. 1994.
3. DeMarco, Tom. Why Does Software Cost So Much? New York: Dorset House Publishing, 1995.
4. Software Program Managers Network. 16 Critical Software Practices For Implementing Performance-Based Management. Ver. 3.0, Arlington, Va.: Integrated Computer Engineering, Inc., 2 Aug. 2000.
5. DeMarco, Tom. Why Does Software Cost So Much? New York: Dorset

Characteristic	Risk Designator	Number of risk events applicable to specific Risk Designator	Number of risk events for Characteristic	Frequency of occurrence relative to all observed risk events (See Note 1)
1. Failure to Apply Essential Project Management Practices.	A	246	480	57%
	I	6		
	J	36		
	C	66		
	K	10		
	L	116		
2. Unwarranted Optimism and Unrealistic Executive Management Expectations.	A	246	344	41%
	B	32		
	C	66		
3. Failure to Implement Effective Software Processes.	G	162	248	30%
	D	15		
	H	26		
	F	45		
4. Premature Declarations of Victory.	L	116	165	20%
	N	46		
	O	3		
5. Lack of Program Management Leadership.	C	66	106	13%
	E	3		
	M	5		
	B	32		
6. Untimely Decision-making.	D	15	63	8%
	E	3		
	F	45		
7. Lack of Proactive Risk Management.	P	4	24	3%
	Q	9		
	R	11		

Note 1: The total number of risk events categorized (841 events) was used as the baseline population of risk events for frequency of occurrence calculations.

Table 1: *Seven Risk Characteristics*

- House Publishing, 1995. 62.
6. Standish Group International. "Chaos." Open Computing Copyright, Mar. 1995 SPC.
 7. Jones, Capers T. Assessment and Control of Software Risks. New Jersey: Prentice Hall, Feb. 1994. 158.
 8. Lister, Tim. "Software Management for Adults." Software Technology Conference, 1996.
 9. Davis, Alan. "Software Lemmingengineering." IEEE Software Sept. 1993.
 10. Humphrey, Watts. "Three Dimensions of Process Improvement: Part I: Process Improvement." CROSSTALK Feb. 1998.
 11. DeMarco, Tom. Why Does Software Cost So Much? New York: Dorset House Publishing, 1995. 66.
 12. Putnam, Lawrence H., and Ware Meyers. Industrial Strength Software, Effective Management Using Measurement. Los Alamitos, Calif.: IEEE Computer Society Press, 1996. 1.
 13. Putnam, Lawrence H., and Ware Meyers. Industrial Strength Software, Effective Management Using Measurement. Los Alamitos, Calif.: IEEE Computer Society Press, 1996. 13.
 14. P.V. Norden. Useful Tools For Project Management, Operations Research in Research and Development. Edited by B. V. Dean. New York: John Wiley & Sons, 1963.
 15. Molt, George. "Risk Management Fundamentals in Software Development." CROSSTALK Aug. 2000.
 16. Boehm, Barry, Raymond Madachy, and Chris Abts. "Future Trends: Implications in Cost Estimation Models." CROSSTALK Apr. 2000.
 17. Hall, Elaine. Managing Risk. Reading Mass.: Addison-Wesley 1997. 5.
 18. DeMarco, Tom, and Tim Lister. Peopleware, Productive Projects and Teams 2nd ed. New York: Dorset House Publishing, 1999. 4.

Did this article pique your interest?

You can hear more from Michael Evans at the Fourteenth Annual Software Technology Conference Apr. 29-May 2, 2002, in Salt Lake City, UT. He will be presenting in Track 6 on Thursday, May 2, at 1:00 p.m.

About the Authors



Michael W. Evans, former owner and president of Integrated Computer Engineering, (ICE) Inc., is now a senior vice president with American Systems Corporation, which recently acquired ICE as a wholly owned subsidiary. He has led more than 250 program-risk assessments of large federal, Department of Defense (DoD), and commercial software acquisition and development projects and is considered an expert in software testing, quality assurance, and configuration management. He is co-founder of the Software Program Managers Network, the driving force behind the DoD's Software Acquisition Best Practices Initiative. His published books include *Principles of Productive Software Management*, *Productive Test Management*, *Software Quality Assurance and Management*, and *The Software Factory*.

Integrated Computer Engineering, Inc.
142 North Central Avenue
Campbell, CA 95008
E-mail: candca@aol.com



Alex M. Abela has 18 years of professional engineering experience, with more than seven years experience as a senior project leader of major Defense programs. He has worked in Defense Research and Development organizations in Australia, United Kingdom, and the United States. His engineering experience spans the disciplines of information technology, electronics, electro-optics, and telecommunications. His interests in information technology include the provision of technical advice on software systems best practices. Currently he is working for the Australian Department of Defense as senior technical specialist in surveillance and reconnaissance systems.

Australian Department of Defense
Land Engineering Agency
DPM-4-32, Defense Plaza
661 Bourke Street
Melbourne, Victoria
Australia 3000
Phone: +61 3 9622 2945
Fax: +61 3 9622 2782



Thomas Beltz serves as executive assistant at American Systems Corporation. He has more than 10 years of experience in Department of Defense and commercial software acquisition, operational test and evaluation (OT&E), and software best practices implementation. At the U.S. Navy OT & E Force, he authored an OT&E Task Schedule with more than 1,000 program-tracking controls to determine system readiness for formal operational evaluation. He co-developed Integrated Computer Engineering's Software Development Capability Evaluation Tool, providing a quantitative assessment of a developer's capability to build software while meeting program life-cycle requirements, and he has participated in more than 25 independent risk assessments of large-scale software development programs.

ICE Directorate EA
Phone: (757) 463-8483, ext. 21
E-mail: thomas.beltz@iceincusa.com

Share Your Ideas With 100,000 People ...

Plus, Get the Chance to Win a Palm Pilot

So you chickened out last year; the 50 foot runway intimidated you. Well, this year is your chance. Come fly your paper airplane at CrossTalk's second annual Paper Airplane Contest at the 2002 Software Technology Conference and show everyone what you are made of. The event will feature valuable prizes, including a palm pilot, door prizes, and free food. You will also get a chance to meet the CrossTalk staff and learn how to become one of the journal's prestigious authors who are read by over 100,000 readers monthly.

2002 CROSSTALK PAPER AIRPLANE CONTEST
April 30, 2002 from 4:45 p.m. to 6:30 p.m.
Software Technology Conference
Salt Palace Convention Center, Salt Lake City, UT