Integrating "Crisis" into Project Management Training

Risk management is, and should be, a hot topic in software engineering, given the frequency and severity of the failed and badly delivered software projects. Though more literature is being published on the topic, the frequency of software delivery problems does not appear to be dropping. In this article we review a few of the best software risk management literature and software tools.

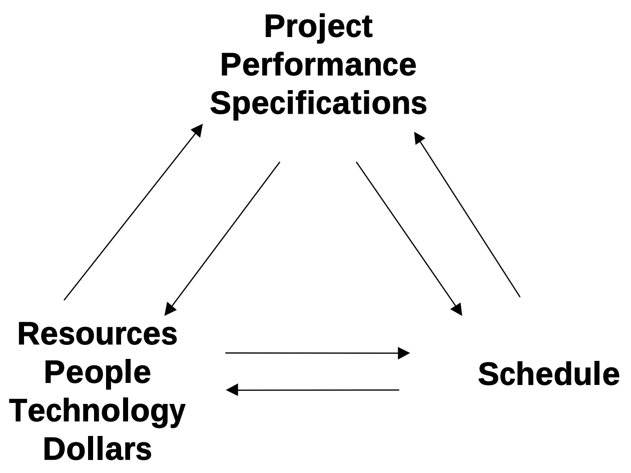
Our conclusion is that the problem does not appear to be in the body of knowledge, but in project management training. Our solution is to introduce "crisis" into the training process. The result is that risk is always present; bad events occur. Risk management is part of every project manager's daily repertoire of project management skills. The key is to train project managers on the determination, assessment, and quick resolution of the unexpected event that creates the disruption.

The Age Old Problem—Software Projects Fail

How many times have you been part of this scenario? Your boss informs you that, with no apparent warning, the software project will be months late, cost more than twice the budget, not meet the required performance, perform an unnecessary task, or be scrapped because it cannot be finished.

The fact that there are software problems is a given. The realization of how poorly we deal with them is of great concern to all of us in the software engineering business. Software problems result because there are always risks involved in the design, creation, and implementation of software. Risks result from incomplete understanding of the project during the initial project planning. Other risks arise from events that occur as projects progress.

Let us consider a simplified scheme of each project. The project management process involves balancing of the three key project components: performance specifications; resources, which include people, technology and dollars, and schedule. Each project manager has the balancing act of assuring that performance can be achieved with resources at the project's disposal and within the time allowed for completion. The project planning process is to carefully review each component to assure compatibility. During this planning process, any change in one of the three key project components requires re-evaluation and realignment of the other two.


Project Performance Specifications
Resources People Technology Dollars
Schedule

The Project—The Balancing Act

Project managers are usually trained with the implicit assumption that the world is fully understood and predictable. They are taught to develop their plans as if project specifications are fixed and well-defined; resources needed are understood, and once-specified, constant; and the agreed-upon schedule will not change. This is a perfect world.

Unfortunately, we in software engineering do not see this perfect world in the projects we encounter. We live in a world where frequent change is required in the scope of a project. Resources we had counted on do not materialize or are inappropriate. The initial schedule is accelerated as the project becomes critical to the firm's success. We must analyze and plan for risk. Substantial risk management literature has developed that discusses ways to manage risk.

Literature on Software Engineering Risk Management

As software engineering practices have matured, there has been great improvement in available tools. The Software Engineering Institute (SEI) at Carnegie-Mellon University has developed the Software Capability Maturity Model® to use as a guide for good practices. The Project Management Institute has published the Body of Knowledge for Project Management that can be utilized in software engineering. While there are several methodologies that can be adapted to any project, the question remains, "Are these methodologies sufficient to offset the risk found in software engineering?" Growing literature on risk management tells us no.

Three publications stand out as central to the body of knowledge for software engineering risk management:

1. SEI's *Continual Risk Management Guidebook* [1].
2. *Tutorial: Software Risk Management* by Barry W. Boehm [2].
3. *Software Engineering Risk Management* by Dale W. Karolak [3].

These authors would generally agree on the following definitions:

Risk—The possibility that a development project incurs a loss as a result of inadequate quality of final system, higher costs, schedule delay, or total failure.

Risk Management—The discipline to discover and eliminate software engineering risk, so that it ceases to threaten the success of a software project.

Boehm's book contains a chapter by Tom Gilb, "Principles of Software Engineering Management" with some provocative quotes on why risk management is important:

The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark office to Carnegie Mellon University

"The risk principle: If you do not attack the risks, they will actively attack you."

"The risk prevention principle: Risk prevention is more cost-effective than risk detection."

"The risk sharing principle: The real professional is one who knows the risks, their degree, and their causes, and the action necessary to counter them, and shares this knowledge with his colleagues and clients."

"The asking principle: If you do not ask for risk information, you are asking for trouble."

"The principle of risk exposure: The degree of risk, and its causes must never be hidden from decision-makers."

SEI's *Continual Risk Management Guidebook* is a helpful volume that provides an extensive and disciplined proactive decision framework to:

1. Assess continuously what could go wrong (risks).
2. Determine which risks are important to deal with.
3. Implement strategies to deal with those risks.

Tutorial: Software Risk Management develops Boehm's framework for software engineering risk management by presenting a set of methods around risk assessment and control.

- Risk Assessment
  - Risk Identification
  - Risk Analysis
  - Risk Prioritization
- Risk Control
  - Risk Management Planning
  - Risk Resolution
  - Risk Monitoring

Software Engineering Risk Management by Karolak uses an interview technique to develop software risk metrics. He presents six risk management activities:

- Risk Identification
- Risk Strategy and Planning
- Risk Assessment
- Risk Mitigation/Avoidance
- Risk Reporting
- Risk Prediction

The SEI, Boehm and Karolak models are quite similar in their general approach. They treat software engineering risk as a planned process. They all have a risk management process that contains a set of repeated steps. These steps focus on risk identification and risk analysis with a clear risk-reporting process, which serves as a means to communicate risks to individuals and organizations involved. Both the SEI and Karolak frameworks contain a risk taxonomy and risk metrics.

We live in a world where frequent change is required in the scope of a project.

While these three frameworks are extremely helpful, they present a more predictable world than the one we encounter in software engineering professional careers. It is almost always the unexpected that creates the crisis and generates the project risk.

We have found that it is impossible to provide a list or description of the risks that we might encounter. And when we have, what turns out to be the killer risk was not on the list: a key employee leaves unexpectedly, the client changes the scope without warning, the technology we thought would work does not, a corporate crisis results in a dramatic reduction in the project's resources, the project schedule is shortened because of changing corporate strategy.

The risk first looms as a crisis that the project team is ill-prepared to handle. The SEI, Boehm, or Karolak frameworks are excellent, but we argue that they are not enough for two reasons. First, many risks will not be definable until a crisis hits. Second, we need to train project managers to be reactive problem solvers as well as thorough risk planners.

Risk Training—Project Planning Simulated Crisis

Introducing crisis into the formal project management training works. Let us move away from the view of project management dealing with a known and predictable world as we introduce individuals to principles of project management. Let us face up to the complexity of the world of software engineering, whether dealing with the development of a new system or the upgrade of an existing one. Risks that create the project crisis are never fully predictable.

Moving to a project management training that introduces the unexpected in the training project or simulation prepares the future project managers for real world reality. Project managers in training are forced to confront the three key project components—performance specifications, resources, and schedule—and rebalance them given the crisis situation. You deal with the actual or potential crisis recognizing that it is a normal and an expected aspect of the project management process. The project management training sessions provide actual training experience on the second aspect of the SEI, Boehm, or Karolak frameworks. You incorporate a crisis in the learning situation that forces the participants to solve an actual problem (risk) that has occurred.

Conclusion

Software engineering contains and will be fraught with risk through the foreseeable future. Risk management provides a useful preplanning perspective. Unfortunately, with the creation of a software engineering project plan, risk frequently cannot be specified during early stages of a project. Learning how to quickly and appropriately respond to an unexpected major alteration in performance specifications, resources, or schedule needs to be instilled into the project manager through training. Training project managers with random crisis situations in course projects works. The training curriculum may be more difficult to create, but the value in developing project managers who can successfully cope and understand risk is worth the effort.◆

References

1. Dorofree, Audrey J.; Walker, Julie A.; Alberts, Christopher J.; Higuera, Ronald P.; Murphy, Richard L.; and Williams, Ray C.; *Continuous Risk Management Guidebook.* Carnegie-Mellon University, 1996.
2. Boehm, Barry W. *Tutorial: Software Risk Management.* IEEE Computer Society Press, 1986.
3. Karolak, Dale W. *Software Engineering Risk Management.* IEEE Computer Society Press, 1996.

About the Authors

Dr. Kenneth Knight is professor of information systems management at Seattle Pacific University. Knight has been active as an author, professor, manager, and consultant in the information systems area for 35 years. His previous academic positions were in the School of Business at The University of Texas at Austin and Stanford University.

Seattle Pacific University
3307 3rd Ave. West
Seattle, Wash. 98119
Voice: 206-281-2906
Fax: 206-281-2733
E-mail: kknight@sup.edu

Darrell Corbin is an associate technical fellow in Companywide Software Engineering at the Boeing Co. with more than 31 years experience in business and engineering information systems. He provides Boeing-wide support in software methodologies and software process improvement. He also conducts reviews and evaluations of IS suppliers, projects, and organizations. He has bachelor's and master's degrees in economics from Washington State University, and is a Certified Computing Professional.

Russ Hamerly is project manager in Distributed Computing Program Integration at the Boeing Co. He has more than 25 years experience in electronic switching, networking, and computing systems. He provides Boeing-wide support in program management, system management, and emerging workstation technologies. He is a subject matter expert for the SRP web site. He has a bachelor's degree from the University of Washington and a master's degree in business administration from the University of Notre Dame.

Roger Cox is a senior manager in engineering information systems at Boeing Commercial Airplanes. He has more than 32 years of information technology experience, including the last 20 years at Boeing. Cox also serves as an adjunct professor of software engineering at the University of Washington and an adjunct professor of information systems in the graduate program in the school of business at Seattle Pacific University. Cox holds bachelor's degrees in engineering physics, mathematics, and computer science, and master's degrees in physics and computer science, is doing doctoral studies in computer science, and holds an advanced management certificate. Cox is a retired Lt. Col. in the Air Force, where he served in various manugacturing, engineering, and information technology organizations.

The Boeing Co.
P. O. Box 3707
Seattle, Wash. 98124-2207
Voice: 206-655-2121
Corbin's e-mail: darrell.s.corbin@boeing.com
Hamerly's e-mail: russell.p.hamerly@boeing.com
Cox's e-mail: roger.l.cox@boeing.com

Coming Events

March 6-8
13th Conference on Software Engineering Education and Training CSEET&T
http://www.se.cs.ttu.edu/CSEET2000

March 6-10
Software Management/Applications of Software Measurement SM/ASM 2000
http://www.sqe.com/smasm/2000/

March 13-17
6th International Conference on Practical Software Techniques
http://www.softdim.com

March 20-22
5th Annual Association for Configuration & Data Management Technical Conference
http://www.acdm.org

March 20-23
12th Software Engineering Process Group Conference
SEPG 2K
http://www.sei.cmu.edu/products/events/sepg/

April 11-14
Infosecurity 2000
http://www.infosec.co.uk/page.cfm

April 18-20
FOSE: Leading-Edge Technology for Leaders in Government
http://www.fedimaging.com/conferences

April 24-28
SEA 2000, held in Canberra, Australian Capital Territory
E-mail for information: johnl@sea-act.com.au

See page 20 for STC 2000, April 30-May 4

June 20-22
Product Focused Software Process Improvement PROFES 2000
http://www.ele.vtt.fi/profes2000

August 28-31
1st Software Product Line Conference SPLC1
http://www.sei.cmu.edu/plp/conf/SPLC.html

wires crossed

Alan Turing, who in the 1930s developed what became known as the Turing Machine, died in 1954. *Influential Men and Women of Software,* in the December issue was, regrettably, in error.