


[About Us](#)
[Consulting Services](#)
[CrossTalk](#)
[Conference](#)
[Resources](#)
[Search](#)

## Software Technology Support Center

[CrossTalk](#)
[Home](#) > [CrossTalk Apr 2002](#) > [Article](#)
**Apr 2002 Issue**

### About CrossTalk

- Mission
- Staff
- Contact Us

### Current Issue

### Subscription

- [Subscribe Now](#)
- [Update](#)
- [Cancel](#)
- [RSS 2.0](#)

### Theme Calendar

### Author Guidelines

### Back Issues

### Article Index

### Your Comments

## CROSSTALK

The Journal of Defense Software Engineering

### Requirements Risks Can Drown Software Projects

Theron R. Leishman, Software Technology Support Center

Dr. David A. Cook, Software Technology Support Center

*Software requirements management is often viewed as a stand-alone task in terms of life-cycle activities. Of course, some of the major risks to project completion are incomplete, inaccurate, or vague requirements. In this article we will present and discuss several requirements risks that may have major impacts on the success of software projects. We will then consider strategies to help mitigate the impact of these requirements risks.*

A few years ago the movie "Overboard" was released. This is a movie about a rich woman (JoAnna) who was accustomed to having everything her own way. The movie begins with JoAnna hiring an uncouth carpenter (Dean) to remodel the closet of her luxurious yacht. Following several unpleasant encounters between the two during the remodeling project, a major confrontation occurs as the carpenter has completed work, and the yacht is preparing to leave port.

While the carpenter is demonstrating the features of his work, the rich, arrogant, JoAnna asks what the closet is made of. In response, Dean indicates that the closet is made of oak. His response pushes JoAnna over the edge. She says that she wanted the closet to be made of cedar. The carpenter responds that if she wanted the closet to be made out of cedar, she should have asked for cedar. He tells her that he would be glad to make the closet out of cedar, but that his estimate would be more than double because he would have to re-do the whole project. To which she responded, "the whole civilized world knows that closets are made of cedar!" She further indicates that she is not going to pay for "his" mistake! The confrontation escalates to the point that she pushes Dean overboard.

This humorous example demonstrates how easily requirements can be confused between the various stakeholders of any venture. Confusion, misunderstanding, and frustration relative to requirements are major risks to the success of any project.

At the 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium in 1999, the results of a study of 1995 Department of Defense (DoD) software spending were presented. A summary of that study is shown in Figure 1. As indicated, of \$35.7 billion spent by the DoD for software, only 2 percent of the software was able to be used as delivered. The vast majority, 75 percent, of the software was either never used or was cancelled prior to delivery. The remaining 23 percent of the software was used following modification [1].

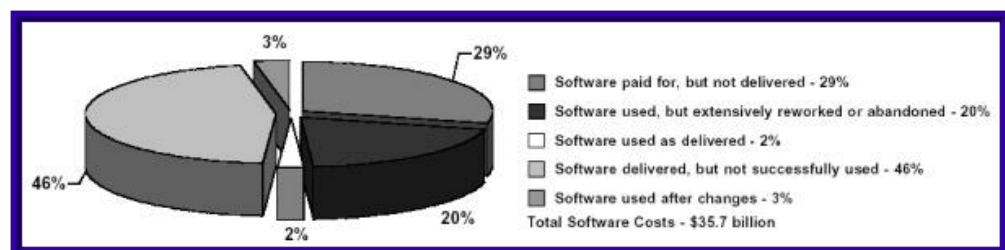


Figure 1: *Findings of a 1995 Department of Defense Software Study*  
(Click on image above to show full-size version in pop-up window.)

A similar study conducted by the Standish Group on non-DoD software projects in 1994 produced very similar results. In over 8,000 projects conducted by 350 companies, only 16 percent of the projects were considered successful. Success in this study was considered software delivered on time and within budget [2].

More recently, an analysis of the data gathered by the Software Engineering Institute (SEI) on 451 Capability Maturity Model® (CMM®) Level 1 CMM-Based Assessments for Internal Process Improvement conducted from 1997 through August 2001 indicates that requirements continue to be a problem. Of the assessments conducted, approximately 95 percent included an assessment of the Requirements Management Key Process Area (KPA). Of these, only 33 percent fully satisfied the Requirements Management KPA [3].

So what does this data mean? Are we as an industry wasting away billions of dollars due to incompetence? What is the reason for this dismal representation of our capabilities? Further research by the Standish Group indicates the following major reasons for the high failure rate in software development:

- Poor requirements.
- Lack of understanding that cost and schedule are engineering parameters based on requirements.
- Lack of understanding and following a process and a life cycle.

Are we all like Dean, the carpenter in "Overboard"? The above studies indicate that the way we define, analyze, and manage requirements is imposing serious risk to the success of our software projects.

## Requirements Risks

### But We Gave You Exactly What You Asked For

Have you ever been disappointed when you received exactly what you asked for? The story is told of an executive who when asked about his satisfaction with a new software application indicated that he hated it. When asked why, he responded, "They gave me exactly what I asked for."

The requirements definition phase of a software project is never the self-contained function implied by many software development life-cycle models. The requirements gathering phase is rather an iterative process. It is not enough to obtain the stakeholder's requirements once and assume that they are correct. By so doing, the risk of giving the stakeholder what they ask for, rather than what they really need, is increased.

### I Know That I Think I Know What Your Requirements Are

This requirement risk can be exemplified by the co-author's (Leishman's) first software project. Being new to the job, he was determined to demonstrate his abilities. Following a short meeting with the project stakeholders, Leishman disappeared into cubie-land to work his magic. A few weeks later, he emerged from his cubbyhole and proudly presented the product of his efforts to the stakeholders. To his dismay, the system did not do what the stakeholders required it to do. They were pleased that their application was delivered on time, but very upset that the application did not do what they required.

This risk is a very common occurrence. It is characterized by not involving project stakeholders throughout the development effort. Typically this requirements risk will not be identified until stakeholder testing or implementation. By not taking necessary steps to assure that we understand the requirements, we are inviting project rework that will result in schedule delays and cost overruns.

### Overboard Assumptions

This is the requirements risk we identified in the movie example in the introduction of this article. There are things in each of our frames of reference that appear to be no-brainers! In the movie, JoAnna indicated that, "The whole civilized world knows that closets are made of cedar." Why then was there a question of the customer's requirements? Because the frame of

reference of the rich and famous was totally different than that of the country hick carpenter.

The risk of assuming that developers and customers have the same thoughts about system requirements is like assuming that we all agree on political or religious issues. As in the case of the carpenter, these issues are often not identified until the customer first sees the application. At this stage of development, there will almost certainly be negative impacts on both project cost and schedule.

### **The Expectations Cloud**

A few years ago, a woman responded to an advertisement to have the carpets in her house cleaned. The advertisement offered three rooms of carpet cleaned for \$69.95. Seeing this as a good deal, she contacted the company and indicated that she would like the three-room special. The representative came to her house and cleaned the carpet in the three rooms identified. Upon completion, the technician presented a bill for \$249. Needless to say, the woman was not a happy camper! When questioned about the bill, the technician indicated that the three-room special applied to rooms with dimensions of 10 feet by 10 feet. The three rooms cleaned were each considerably larger than this.

So, who was at fault in this carpet-cleaning situation? Was it the woman's responsibility to assure that there was agreement to the expectations prior to the work being completed? Or was it the responsibility of the service provider to assure that he met the woman's expectations? Regardless of where the responsibility for clarification belonged, the woman was a dissatisfied customer!

There are numerous software project examples of stakeholder expectations not being resolved prior to the project beginning. Like the carpet example, if the customers' expectations are not met, future services will not be requested.

### **The Never-Ending Requirements Story**

The world we live in is rapidly changing. The business, economic, political, and military environments are all changing and fluctuating daily based on world conditions. Technology is likewise changing at a very rapid rate. In this environment, it is unrealistic to assume that software requirements are not going to change during the development process.

The risk of requirements changing is that the changes will spiral out of control and prevent an application from ever being completed. The never-ending requirement story is also one of the never-completing software project. An additional risk of constant change is the lack of common requirements understanding by the various stakeholders during the various iterations of requirements. The result is confusion, chaos, misunderstanding, and software either never being completed, or if completed never being used.

### **I Don't Know What I Want, but I'll Know It When I See It**

There are also those occasions when stakeholders are not interested in clearly defining their requirements. These stakeholders assume they have the luxury of having developers play the development game over and over until they get something they like.

The risk of allowing this attitude to exist in the stakeholder environment is that valuable development resources are wasted playing a guessing game. Every iteration expended and left floundering in the dark to arrive at a solution that meets the stakeholders' unknown requirements increases development costs and extends the project schedule.

A further risk with this attitude is that the software project is not perceived to belong to the stakeholders. They remain removed from the project, do not accept ownership, and thus increase the risk of project failure.

### **Rapid Development Requirements Risks**

In an attempt to increase the speed of software development, various rapid development approaches have been devised. The use of Rapid Application Development, Unified Modeling Language Use Cases, and Extreme Programming are examples of new approaches that have been taken to speed software development.

When conducted properly, each of these approaches includes a process of requirements

analysis. Often these new approaches have sought newer, better ways to document, verify, and track stakeholder requirements. The risk associated with these approaches is the same old temptation to cut corners when conducting requirements analysis that we often find in more traditional approaches. Often, developers who are not properly trained in these approaches assume that "rapid" means incomplete or haphazard. Some do not recognize the need for adequate requirements analysis and will eliminate or minimize the requirements analysis process. This is a tremendous risk to project success.

### **Failure to Recognize that Faulty Requirements Represent a Risk**

According to the SEI Software Capability Maturity Model® (SW-CMM®), "Requirements management involves establishing and maintaining an agreement with the customer on the requirements for the software project. The agreement forms the basis for estimating, planning, performing, and tracking the software project's activities throughout the software life cycle [4]." Years of experience have revealed that errors occurring in the requirements stage of the development process turn out to be the most difficult and costly to fix.

The Software Technology Support Center has, upon the request of various program managers, conducted Independent Expert Program Reviews. These reviews are conducted to assist program managers in determining the wellness of their programs and evaluating areas of concern to the program. A common finding in many of these reviews is that requirements elicitation, analysis, and management is being conducted in an inefficient manner. In several of these cases, requirements inadequacies have proven to be a major contributor to the program being behind schedule and over budget.

It is essential to understand that requirements, or more appropriately the lack of adequate requirements, can be a significant risk to the success of any project. The soundness of the organization's requirements management process should be taken into consideration when evaluating the risk that requirements may have on the success of your project.

### **Can't You Read My Mind?**

Failure to document requirements in a format that promotes clear, complete, and comprehensive understanding is a serious risk to project success. In his article, "When Telepathy Won't Do: Requirements Engineering Key Practices [5]," Karl Wiegers indicates that the most essential and yet often neglected practice is to write down, or document, the requirements. He goes on to indicate that requirements should be documented in some acceptable, structured format as they are gathered and analyzed.

After all, is not the purpose of the requirements process to communicate a shared understanding of what the system requires among the project stakeholders? The need to communicate requirements is directly connected with the need to document requirements. If requirements are not documented in some manner, it is impossible for multiple individuals to come to a common understanding and agreement of the requirements.

## **Strategies to Mitigate Requirements Risks**

### **Develop and Follow Sound Processes and Procedures**

A software process can be defined as a set of activities, methods, practices, and transformations that people employ to develop and maintain software and the associated products [4]. The problem comes when the process that I follow and the process others in the organization follow is not the same. The result is a lack of consistency in the way software is developed and maintained. This lack of consistency leads to confusion, misunderstandings, development delays, and cost overruns.

An important step to mitigating requirements risks is for the organization to develop and strictly follow sound processes and procedures relative to requirements engineering. These processes and procedures should include direction to developers in the following areas:

- Requirements elicitation.
- Requirements analysis.
- Documentation of requirements.
- Requirements verification, review, and approval.
- Configuration control of requirements.
- Requirements traceability.

The organization should also ensure that roles and responsibilities relative to these processes and procedures are clearly defined. In one aerospace software development organization we know, the application development manager went to great lengths to impress upon the software project leaders that the ultimate success of each development project rested upon their heads. The responsibility for resolving assumptions, ambiguities, and clarifying stakeholder expectations rests upon the developers.

### **Incorporate Requirements into All Software Life Cycles**

By now, the need and value of having organizationally accepted software lifecycle models and methods should be well established. From our research and knowledge of various life-cycle models, they all include a requirements analysis, requirements management, or stakeholder requirements phase.

By assuring that the life cycle(s) approved for usage within the organization require proper levels of requirements administration, the risk associated with projects relative to requirements will be reduced.

### **Provide Training to Those Responsible for Requirements Management**

Requirements elicitation, analysis, documentation, verification, and maintenance are not simple tasks. The ability to facilitate the elicitation of requirements and follow the process through to completion requires knowledge and skill. Those within the organization who are responsible for assuring that requirements are managed, must receive the training and mentoring necessary to provide them with the ability to fulfill this responsibility.

### **Require User Involvement**

In their CHAOS Report of 1995, the Standish Group indicated that information technology (IT) projects fail because they lack user involvement, follow incomplete requirements and specifications, and experience confusion caused by changing requirements and specifications. The lack of adequate user involvement is a virtual guarantee of project failure.

In the article, "13 Common Objections Against User Requirements Analysis, and Why You Should Not Believe Them [6]," the author looks at 13 excuses used for not involving users in the development of Web-based projects. The conclusion drawn is that user involvement is imperative to the success of IT projects even in the rapidly changing web environment.

Users are the primary stakeholders in most software projects. To assume that the primary stakeholder can be eliminated and have the project succeed is approaching lunacy. Yet as indicated in the study conducted by the Standish Group, lack of user involvement in requirements analysis and verification is the root cause of many development project problems.

The need for user involvement is imperative. By requiring user involvement in the requirements process, the risk to the project of inadequate requirements is greatly reduced. This mitigation strategy combined with having and following sound processes and procedures, which are also supported by defined software life cycles, will greatly reduce requirements risks.

### **Always Document Requirements**

The need to reach a common understanding of requirements among key project stakeholders is vital to the success of a software project. To reach common understanding it is essential that requirements be documented. Any text on software requirements will indicate that good requirements' characteristics include the following:

- Correct.
- Complete.
- Consistent.
- Unambiguous.
- Verifiable.
- Understandable.
- Traceable.
- Modifiable.

Historically, requirements have been documented in a System Requirements Specification

(SRS) or other similar document. In recent years requirements documentation has taken on various forms. Today there are three basic forms of documentation used for requirements: text based, box based, and graphics based.

Text-based documentation relies on formally defined language to describe system requirements. This approach has been criticized as being old fashioned, slow, and subject to various interpretations based on the background of the various stakeholders.

Box-based documentation uses geometric symbols to represent various aspects of the system requirements. This approach is designed with the software engineer in mind and is generally more comfortable for the software engineer to follow and understand. This approach is traditionally more difficult for end users to understand because of the learning curve associated with this type of documentation.

Graphic-based documentation is the more recent of the three documentation forms. It was developed to support object-oriented development and design techniques. This method uses geographic symbols to represent the actual objects within a system. As with the box-based methods, graphic-based documentation is tailored more toward the developer than to the end user's understanding.

We recommend that some combination approach be adopted. We have experience using a combination of graphics and text. This approach helps the users insert themselves into the process or requirement being documented. It also is easily understood by the developers and serves as a useful tool to both elicit and validate requirements.

Various programming methodologies such as Extreme Programming and the Unified Systems Development Process are typically considered design and coding aids. These approaches recommend proper requirements gathering, analysis, and validation be conducted during software development and maintenance. These methodologies document requirements using models, diagrams, and text in an environment of heavy stakeholder involvement. These requirements can and should be maintained during various iterations of development and their traceability maintained through the entire development process.

### **Validate Software Requirements**

Requirements validation is an essential step to ensure that requirements are properly understood and documented. This mitigation strategy goes hand in hand with the need for dedicated stakeholder involvement. As requirements are documented, the stakeholders should validate them. Often the act of requirements validation will uncover requirements issues that can be discovered in no other way.

### **Hold Formal Requirements Reviews**

It has consistently been proven that it costs more to correct requirements errors that are discovered later in the life cycle. For example, a requirement error caught during design might cost four times more to correct than if the error had been discovered and corrected during the requirements phase itself [7]. Studies indicate that document reviews greatly reduce the errors in critical documents.

Errors, inconsistencies, ambiguities, and confusion can be greatly reduced by holding formal reviews of software requirements documents. Teams performing requirements management activities should be trained in sound review methods. Formal reviews will greatly improve the quality of software requirements documents.

### **Strictly Manage Requirements Changes**

Strategies exist to help manage requirements "creep." Such strategies consist of good configuration management, formal reviews of change requests, and a formal change control process. Requirements creep should be around 1 percent per month. In fact, creep of more than 2 percent a month is probably a sure sign of a project that will never reach completion. Without sound strategies for managing change, a project will fail.

Even with sound strategies, stakeholders must be aware of the high cost and high risk of change. For the good of the project, some changes are simply too expensive or too difficult. Such changes must be postponed until after a version of the product is successfully delivered.

## Summary

Requirements management is critical to the success of any software development or acquisition project. It requires the ability to deal with stakeholders of various backgrounds with various goals, interests, and objectives. By their very nature, there are risks associated with the elicitation, analysis, and validation of requirements. If not given proper attention, these requirements' risks can push software projects overboard and result in software projects drowning!

By recognizing the potential impact of these requirements' risks, steps can be taken to turn these risks into strengths. Instead of requirements being the source of problems, a disciplined software requirements management process can help to assure the success of your software projects.

## References

1. Jarzombek, Stanley J. "The 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium (JAWS S3)." Proceedings, 1999.
2. The Standish Group International, Inc. *The CHAOS Report*, 1994.
3. Software Engineering Institute. "Process Maturity Profile of the Software Community." Mid-year Update, Aug. 2001.
4. Software Engineering Institute. *The Capability Maturity Model Guidelines for Improving the Software Process*. Boston: Addison-Wesley, 1994.
5. Wiegers, Karl E. "When Telepathy Won't Do: Requirements Engineering Key Practices." *Cutter IT Journal* May 2000.
6. D'Hertefeldt, Sim. "13 Common Objections Against User Requirements Analysis, and Why You Should Not Believe Them." *Interaction Architect.com* 9 June 2000.
7. Boehm, Barry, and Wilfred J. Hansen. "The Spiral Model as a Tool for Evolutionary Acquisition." *CrossTalk* May 2001.

## Additional Reading

1. Dorfman, Merlin. "Requirements Engineering." *SEI Interactive* Mar. 1999.
2. Kar, Pradip, and Michelle Bailey. "Characteristics of Good Requirements." INCOSE Symposium, 1996.
3. Thomas, Bill. "Meeting the Challenges of Requirements Engineering." *SEI Interactive* Mar. 1999.
4. VanBuren, Jim, and Dr. David A. Cook. "Experiences in the Adoption of Requirements Engineering Technologies." *CrossTalk* Dec. 1998.
5. Wiegers, Karl E. *Software Requirements*. Microsoft Press, 1999.
6. Wiegers, Karl E. "Karl Wiegers Describes 10 Requirements Traps to Avoid." *Software Testing and Quality Engineering* Jan./Feb. 2000.

---

## About the Authors



**Theron R. Leishman** is a consultant currently on contract with the Software Technology Support Center at Hill Air Force Base, Utah. Leishman has 18 years experience in various aspects of software development. He has successfully managed software projects and performed consulting services for the Department of Defense, aerospace, manufacturing, health care, higher education, and other industries. This experience has provided a strong background in systems analysis, design, development, project management, and software process improvement. Leishman has a master's in business administration from the University of Phoenix. He is a Level II Certified International Configuration Manager (CICM) by the International Society of Configuration Management (ISCM), and is employed by TRW.

Software Technology Support Center  
7278 4th Street  
Bldg. 100 G19  
Hill AFB, UT 84056

Phone: (801) 775-5738  
Fax: (801) 777-8069  
E-mail: [theron.leishman@hill.af.mil](mailto:theron.leishman@hill.af.mil)



**David A. Cook, Ph.D.**, is the principal engineering consultant, Shim Enterprises, Inc. He is currently assigned as a software-engineering consultant to the Software Technology Support Center at Hill Air Force Base, Utah. Dr. Cook has more than 27 years of experience in software development and software management. He was formerly an associate professor of computer science at the U. S. Air Force Academy (where he was also the department research director) and also the deputy department head of the Software Professional Development Program at the Air Force Institute of Technology. Dr. Cook has published numerous articles on software process improvement, software engineering, object-oriented software development, programming languages, and requirements engineering. He has a doctorate degree in computer science from Texas A&M University, and he is an authorized Personal

Software Process instructor.

Software Technology Support Center  
7278 4th Street  
Bldg. 100  
Hill AFB, UT 84056  
Phone: (801) 775-3055  
Fax: (801) 777-8069  
E-mail: [david.cook@hill.af.mil](mailto:david.cook@hill.af.mil)

---

® Capability Maturity Model, CMM, Software Capability Maturity Model, and SW-CMM are registered in the U.S. Patent and Trademark Office.

---

## Did this article pique your interest?

You can hear more from these authors at the Fourteenth Annual [Software Technology Conference](#) Apr. 29-May 2, 2002, in Salt Lake City, UT. They will be presenting in Track 7 on Wednesday, May 1, at 10:00 a.m.



[Privacy and Security Notice](#) · [External Links Disclaimer](#) · [Site Map](#) · [Contact Us](#)

Please [E-mail](#) or call 801-775-5555 (DSN 775-5555) if you have any questions regarding your [CrossTalk subscription](#) or for additional STSC information.

**Webmaster:** 517th SMXS/MDEA, 801-777-0857 (DSN 777-0857), [E-mail](#)

**STSC Parent Organizations:** [309SMXG Ogden Air Logistics Center, Hill AFB](#)