

Identifying and Controlling Requirements Risk

David Gelperin
ClearSpecs Enterprises

The Problem

Requirements development is hard.

- ◆ Fred Brooks is still right: **The hardest single part of building a software system is deciding precisely what to build**
- ◆ Arthur Schlesinger is right: ... **the possibilities of the future are more various than the human intellect is designed to conceive**
- ◆ Specification is hard: **Ambiguity is the rule, not the exception. The devil's in the details**
- ◆ Checking is harder: **"Best practice" technical reviews miss 2/3 of the defects** - Gilb

Requirements development is **very** hard.

- ◆ Some measures
 - ◆ types of info (~50)
 - ◆ discovery techniques (~20)
 - ◆ checking techniques (~50)
 - ◆ risk management tactics (~60)

This means there are more than 10^{50} requirements development strategies

- ◆ Learning and change happen
- ◆ Human factors get in the way

Major requirements-related problems remain, especially on large-scale projects, even when projects use most of the basic Requirements Development (RD) and management practices such as: (1) identifying enterprise needs and opportunities (2) identifying and involving critical stakeholders (3) defining vision and scope (4) making a business case (5) discovering, specifying, prioritizing, and verifying requirements, and (6) managing versions and changes.

Some (e.g., in the agile community) believe that large-scale requirements cannot be done effectively – and they may be right. History provides strong evidence for this view. A different interpretation is that *just doing the basics is not enough*.

A Solution

Sound RD practices must be supplemented with Requirements Risk Management (RRM). There are tactics for preparing, staffing, and planning for RRM as well as avoiding, minimizing, monitoring, mitigating, and compensating for requirements-based problems.

RRM tactics include: defect and root-cause profiles, cross-functional teams, rich definitions, patterned specs, visualization, incremental reviews, frequent demos of understanding, and requirements retrospectives. These tactics supplement your current development methods and can be used in any combination to manage your requirements risk.

ClearSpecs Enterprises

www.clearspecs.com

version 37

Assumptions

1. Requirements development is not a phase.
2. Requirements development (discovery) is hard.
3. Risks inherent in politics, economics, technology and human nature meet in requirements.
4. If your requirements are not already invalid, they will be soon.
5. Requirements can be dangerous.
6. Perfect, large-scale (more than 30 pages) requirements are unlikely.
7. One or two risky requirements (early delivery, safety criticality, high reliability) can make otherwise safe projects risky.
8. While requirements are important, they are but one area of project risk.
9. With about 50 types of information, about 20 discovery technique, and about 50 checking techniques, there is no single (context-free) "best practice" for requirements development.
10. The "best strategy" for requirements development has two steps:
 - a. Develop a process to fit your project context
 - b. Supplement with safety tactics, if requirements are risky.
11. "Risk management is project management for adults." – DeMarco & Lister
12. Improving your requirements process will not be enough to control requirements risk.
13. Requirements safety tactics can supplement any system development style (waterfall, spiral, agile, lean, developer-driven, yours).
14. There is no "silver bullet" (single safety tactic) for requirements risk.
15. Our list of requirements safety tactics is broad but not complete; add tactics that work for you.
16. Some of our safety tactics won't help you.
17. Other tactics (asking a subcontractor to manage the risk, classifying certain risks as outside your responsibility) won't help anyone.
18. Making humans responsible for safety is a risky strategy – many still don't buckle up.

Safer Reqts in a Nutshell

<u>Identify</u>	and	<u>Control</u>	Reqts Risk
risk "to" reqts		safety tactics	
Application		Monitor reqts and project feasibility	
Stakeholder		Avoid reqts-related mistakes	
Process		Mitigate impact of reqts-based problems	
Resource		Minimize reqts-based problems	
Complexity		Monitor reqts and risk status	
Change		Staff for safer development	
risk "from" reqts		Plan for reqts risk mgmt	
Implementation		Prepare for reqts risk mgmt	
Defect		Compensate for reqts-based problems	

It's just **organized** common sense: 95% existing 5% new

Requirements Risks

Risks **to** requirements

Application

- unfamiliar
- complex
- dangerous

Stakeholder

- incorrect assumptions
- incorrect or insufficient knowledge
- insufficient skills or experience
- insufficient involvement
- disruptive behavior

Process

- scope too narrow or too broad
- inadequate discovery
- incorrect derivation
- insufficient attention to nonfunctional reqts
- confusing responsibilities
- context independent (i.e. rigid) tasks
- ineffective communication or negotiation
- too much or too little specification
- inadequate verification and validation
- creeping scope
- ineffective change or info management
- inadequate defect data collection

Resource

- critical stakeholders overlooked, unavailable, or underinvolved
- arbitrary schedule as hard requirement
- inadequate time for analysis and learning
- insufficient tool support

Complexity

- social (stakeholder group)
- application (problem)
- design (solution)

Change

- individual reqts unstable
- scope expands or refocuses
- rate of change does not decrease rapidly enough
- significant change occurs late

Risks **from** requirements

Implementation

- misinterpretation
- innovation
- design
- integration
- schedule
- budget

Defect

- budget and schedule overruns
- system failures
- poor interfaces
- inappropriate architecture
- unnecessary complexity
- unnecessary functionality
- project cancellation
- system abandonment
- lost opportunities
- unnecessary complexity

Requirements Risk Management

Categories of Tactics

- 1. Monitor Requirements and Project Feasibility**
Projects become “famous failures” by promising much, consuming resources, and delivering little or nothing. This category focuses on avoiding this type of fame.
- 2. Avoid Requirements-related Mistakes**
This category focuses on deepening and validating stakeholder understanding and developing an effective RD process to reduce the likelihood of mistakes.
- 3. Mitigate Impact of Requirements-based Problems**
Requirements will conflict and be defective, despite your best efforts. This category focuses on reducing the impact of these problems.
- 4a. Minimize Communication Problems**
Clear project communication is one of the core challenges of system development. Natural language is a major problem, because it is so natural. This category describes alternatives and supplements to blocks of text.
- 4b. Minimize Other Problems**
In addition to communication, there are other problems that can be reduced with specific minimization tactics. This category describes these added tactics.
- 5. Monitor Requirement and Risk Status**
Requirements and their related information (e.g., defects) must be monitored to detect expected and unexpected risk. This category describes tactics that provide this monitoring.
- 6. Staff for Safer Development**
People, their involvement, knowledge, and behavior make all the difference. This category focuses on getting the right people, with the right perspectives and skills to do the job right.
- 7. Plan for Requirements Risk Management**
To get risk management off to a solid start, you need to understand what bad things might happen. This category deals with creating effective strategies risk management.

Categories of Tactics (continued)

8. Prepare for Requirements Risk Management

Some tactics (e.g., acquiring tools) involve the creation, outside of a development project, of a foundation that enables other safety tactics (e.g., identify imprecise information) to be more effective during development. This category describes these foundational tactics.

9. Compensate for Requirements-based Problems

Because misunderstandings will occur and mistakes will be made, some design, implementation, and test work will need to be redone. This category describes this type of tactic.

Requirements Risk Management (RRM) Tactics

1. Monitor Feasibility

- Reconfirm reality and satisfiability of stated needs
- Triage and prioritize reqts
- Manage customer expectations
- Reconfirm project feasibility or reduce scope
- Identify and resolve conflicts early
- Commit incrementally

2. Avoid Mistakes

- Include stakeholders with a deep understanding of the solution domain
- Study solutions to similar problems
- Have developers maintain similar applications
- Immerse stakeholders i.e., customers in system dev and developers in customer activities
- Provide training in domain concepts and terminology
- For vague requirements, identify alternative solutions and their costs
- Prototype unfamiliar functions and Interfaces
- Have experienced technical writer author larger specs
- Until the project ends, have each stakeholder aggressively try to learn and share understanding
- Elicit meta-requirements
- Customize a "just enough" reqts strategy

3. Mitigate Impact

- Require frequent demos of understanding e.g., early test design
- Carefully analyze results of each increment
- Decouple vision from build
- Subdivide project scope
- Build in multilane cycles
- Use technologies that match a solution

4a. Minimize Comm Problems

- Maintain a climate of personal safety and respect
- Use dialogue mapping
- Isolate known details
- Use rich definitions
- Replace "magic numbers" with derived values
- Identify user types and create user personas
- Clarify with examples and measures
- Structure information with spec patterns - avoid text blocks
- Picture reqts info
- Capture context

4b. Minimize Other Problems

- Identify minimal sets of marketable features
- Focus on quality and environmental reqts early
- Identify impacts and mitigation strategies
- Analyze benefits, risks, priority, and cost of proposed reqts and changes

5. Monitor Requirements and Risk

- Monitor reqts instability and growth along with their major causes and project impacts
- Monitor accuracy of assumptions, expectations, and priorities
- Monitor stakeholder participation
- Use elicitation workshops to find issues early
- Incrementally review long specs
- Identify unclear, incorrect, missing and unnecessary info
- Estimate defect risk
- Trace and analyze reqt derivations and links to resulting workproducts
- Verify satisfaction arguments of derived requirements
- Track reqts defects
- Analyze and classify reqts defects and root causes

Requirements Risk Management (RRM) Tactics (continued)

6. Staff

- Build cross-functional team
- Supplement with outside knowledge, skill, and experience
- Replace underperformers and disruptors

7. Plan

- Brainstorm risks and tactics
- Identify risk indicators and root causes
- Customize, carry out, and monitor RRM strategy

8. Prepare

- Systematize change mgmt and reqts defect and failure tracking
- Provide tools supporting safer reqts
- Develop reqts defect and root cause profiles (occurrence and impact)
- Conduct reqts retrospectives

9. Compensate

- Estimate & track reqts-based rework
- Adjust for surprises – both good & bad

Using the Tactics Checklist

1. Remove tactics already a part of your reqts development, system development, or project mgmt process.
2. Identify tactics for permanent integration into your reqts development, system development, or project mgmt process. After integration, remove these from your checklist.
3. Remove tactics that do not apply to your application domain or are incompatible with your culture.
4. Add optional tactics you have used effectively for controlling reqts risk.
5. Consider remaining tactics for inclusion in your requirements risk management strategy at the beginning of and during each project after reqts risks have been recognized.

Requirements Risk Management References

All papers available at www.clearspecs.com

Reqs Risk Management

Gelperin, David "Controlling Requirements Risk"
McBreen, Pete "The Myth of Risk Management"

Triage and Prioritize Reqs

Davis, Alan **Just Enough Requirements Management** (Chapter 3) Dorset House 2005
Firesmith, Donald "Prioritizing Requirements"
Wiegers, Karl **Software Requirements** (Chapter 14) Microsoft Press 2003

Manage Customer Expectations

Karten, Naomi **Managing Expectations** Dorset House 1994

Resolve Conflicts Early

Dana, Daniel **Conflict Resolution** McGraw-Hill 2001

Incremental Commitment

Boehm, Barry "Project Termination Doesn't Equal Project Failure"
Boehm, Barry and Lane, Jo Ann "Using the Incremental Commitment Model to Integrate
System Acquisition, Systems Engineering, and Software Engineering"
Gelperin, David "The Only Winning Move"

Customize "just enough" Strategy

Gelperin, David "A lot or a little"
Andrea, Jennitta "Agile Requirements for Stepsister Projects"

Frequent Demos

Cuellar, Roland "Test-Driven Requirements"

Prototype

Baskerville, Richard and Stage, Jan "Controlling Prototype Development through
Risk Analysis"
Wiegers, Karl **Software Requirements** (Chapter 13) Microsoft Press 2003

Subdivide Scope

Cockburn, Alistair "Incremental and Iterative Development"

Requirements Risk Management References - continued

All papers available at www.clearspecs.com

Climate of Personal Safety

Smith, Steven "Safety Check"

Wentzel, Paul-Roux et. al. "Human Requirements Engineering"

Use Dialogue Mapping

Conklin, Jeff **Dialogue Mapping** John Wiley 2006

Use Rich Definitions

Gelperin, David "What's It Mean?"

Gelperin, David "Let Me Be Perfectly Clear"

Gelperin, David "Rich Glossary Template for Word"

Create User Personas

Cooper, Alan **The Inmates are Running the Asylum** (Chapter 9) SAMS 1999

Structure Spec Info

Gelperin, David "Clearly Specifying System Reqts"

Gelperin, David "Using Reqts Spec Patterns"

Gelperin, David "Precise Use Cases"

Gelperin, David "Modeling Alternative Courses in Detailed Use Cases"

Gelperin, David "Action Contracts"

Gelperin, David "Quality Specs"

Simmons, Eric "Quantifying Quality Requirements Using Planguage"

Withall, Stephen **Software Requirement Patterns** Microsoft Press 2007

Picture Reqts Info

Lengler, Ralph Eppler, Martin "Towards a Periodic Table of Visualization Methods for Mgmt"

Wieggers, Karl **Software Requirements** (Chapter 11) Microsoft Press 2003

Winant, Becky "Visual Requirements"

Capture Context

Attwood, Katrina Kelly, Tim McDermid, John "The Use of Satisfaction Arguments for Traceability"

Requirements Risk Management References - continued

All papers available at www.clearspecs.com

Sets of Marketable Features

Denne, Mark & Cleland-Huang, Jane **Software by Numbers** Prentice Hall PTR 2003

Quality Reqts

Harty, Julian Evans, Isabel Reid, Stuart "Know What's at Stake"

Workshops

Gottesdiener, Ellen "Collaborate for Quality"

Gottesdiener, Ellen **Requirements by Collaboration** Addison-Wesley 2002

Wood, Jane & Silver, Denise **Joint Application Development** John Wiley 1995

Review Specs

Gelperin, David "High-Impact Inspections"

Wieggers, Karl **Peer Reviews in Software** Addison-Wesley 2002

Analyze Specs

Gelperin, David "Detecting Missing Requirements"

Gelperin, David "Detecting Existing Defects"

Estimate Defect Risk

Gilb, Tom "Specification Quality Control"

Verify Derived Requirements

Hammond, Jonathan et. al. "Will it work?"

Cross-Functional Teams

Parker, Glenn **Cross-Functional Teams** Jossey-Bass 2003

Maxwell, John **The 17 Indisputable Laws of Teamwork** Thomas Nelson 2001

Gelperin, David "Developing a Requirements Team"

Brainstorm Risks

Beasley, Mark & Jenkins, Gregory "A Primer for Brainstorming Fraud Risks"

Ambler, Scott "Overcoming Requirements Modeling Challenges"

Firesmith, Donald "Common Requirements Problems"

Leishman, Theron Cook, David "Requirements Risks Can Drown Software Projects"

Wieggers, Karl **Software Requirements** (Chapter 23) Microsoft Press 2003

Requirements Risk Management References - continued

All papers available at www.clearspecs.com

Provide Tools for Safety

Gelperin, David "TEKchecker User Guide"

Gelperin, David "StyleWriter and TEKchecker Movie"

Analyze Root Causes

Card, David "Learning from Our Mistakes with Defect Causal Analysis"

Weller, Ed "Stop the Insanity"

Wiegers, Karl **Software Requirements** (Appendix C) Microsoft Press 2003

Profile Reqts Defects and Root Causes

Hayes, Jane Huffman "Building a Requirement Fault Taxonomy"

Lutz, Robyn & Mikulski, Ines Carmen "Ongoing Requirements Discovery in

Reqts Retrospectives

Derby, Ester and Larsen, Diana **Agile Retrospectives** Pragmatic Bookshelf 2006

Kerth, Norman "The Ritual of Retrospectives"

Kerth, Norman **Project Retrospectives** Dorset House 2001