Dealing with Requirements

Caroline Pepa


At some time, we’ve all had to deal with customers who didn’t understand the requirements process. Perhaps they changed their minds about what they had agreed to and don’t see why that would be a problem. Anyone who’s dealt with software requirements will appreciate More About Software Requirements.

We wonder how much time we should devote to developing requirements and whether we should just forge ahead with design based on what we already have. Should we include more design in the requirements document? How do we know we correctly understand the requirements when the customer can’t provide them? How do we estimate how long it will take to develop the product or estimate its size? Who are the stakeholders, anyway? More About Software Requirements addresses these issues.

Business concerns

Several studies describe development organizations’ failure to deliver on-time, on-budget software projects. Many determined that having better requirements defined early in the development cycle would have made a significant difference in the quality of the product delivered. Managing the development effort to control chaos and deliver products on time must become the focus of project leaders and program managers. Author Karl Wiegers offers helpful suggestions for building estimates on the basis of the software requirements and their attributes.

Identifying and involving stakeholders

Wiegers defines a stakeholder as “an individual or group who is actively involved in the project, who is affected by the project, or who can influence its outcome,” and that definition covers a lot of territory. Somehow, requirements analysts must foster communication among the stakeholders. The analysts must determine early on whom to empower to make decisions and resolve conflicts among the various stakeholders.

In fact, Wiegers states as a cosmic truth that “the interests of all the project stakeholders intersect in the requirements process.” Two other thought-provoking cosmic truths are “Customer involvement is the most critical contributor to software quality” and “The customer is not always right, but the customer always has a point.”

Customer involvement can take several forms, but Wiegers points out that customers should be involved in more than just “a workshop or two early in the project.” The more involved they are, the less likely they’ll be to throw darts when problems arise—keeping customers involved helps them be part of the solution when things aren’t going well. Another of his cosmic truths is that the customer might have a point that should be heard, but the requirements analyst shouldn’t always accept a customer request without trying to understand the rationale behind it. Customers might not realize that the commitments being demanded are impossible; they might provide solutions disguised as requirements; or they might fail to communicate business rules and other constraints.

Good (if not new) advice

Determining a project’s scope and where to draw the proverbial line in the sand aren’t new issues, but Wiegers’ suggestions that might help project managers and requirements analysts face those issues. You won’t find too many surprises, but the book has worthwhile, thought-provoking suggestions.

Many of Wiegers’ suggestions refer to his larger requirements work, Software Requirements (2nd ed., Microsoft Press, 2003), which
A Simple 10-Step Damage Control Process for Runaway Projects

Naseem Mariam


Catastrophe Disentanglement gives a crisp 10-step guide to getting runaway projects on track. Although the subtitle mentions software projects, you can use the same process with a project that’s over budget or off schedule in any domain or field.

Author E.M. Bennatan spent many years as a senior director at Motorola, developing large software systems and leading multinational design centers. On the basis of this hands-on management experience, he derives his process from a wealth of case studies, scenarios, and live projects.

Explaining the process

Bennatan organizes his 10-step process into four parts, according to who owns the tasks:

- **Launch the process.** Top management must stop the project and assign an evaluator (steps 1 and 2).
- **Evaluate the status.** The evaluator must assess the project and the team as separate entities (steps 3 and 4).
- **Introduce changes.** The original team defines the minimum achievable goals and steps to rebuild itself (steps 5–7).
- **Prepare to resume.** The rebuilt team analyzes the risks, revises the plan, and creates an early warning system (steps 8–10).

This organization makes a lot of sense when you look at who usually has the authority to execute these steps effectively. However, if Bennatan had combined steps 5 and 6, he could have added the epilogue (which puts the pieces together) as an important final step.

Some praise

The book’s beauty is in the first four steps, which are new ways of thinking about an age-old problem. To stop what you’re doing is tough; to differentiate between a runaway project and a controllable, temporary crisis is even tougher. So, it pleased me to see that Part 2 (Evaluate the status) evaluates the project and the team members as separate entities. This thinking is crucial to an organization’s health, yet so rare!

The book has several small touches that I value including “tips before you proceed” (chapter 1), exercises (chapter 2), three ways to better evaluate large projects (that is, as reduced scope, more time, and more people), and shaded advisory boxes.

Each chapter has a “What can go wrong (and what you can do about it)” section and a summary. An experienced manager who’s short on time could glance through these sections to great benefit. Others can use these sections as a refresher.

And some suggestions

The early warning system’s five elements (see figure 12.1 on page 212) are too generic. Examples and stories speak louder than words, and it’s disappointing that Bennatan didn’t include an appendix with templates, sample metrics, and reports. I would have liked to see the evaluator’s report and checklists for the evaluator and the rebuilt team.

This book is good reading for middle and top management (and their advisors) because they can influence whether a project moves forward. A junior engineer or line manager might not be able to drive the action toward the first five steps. Furthermore, experienced managers and decision makers will benefit greatly from this book, whereas it could overwhelm a young engineer who is caught up in a badly managed project.

Readers interested in learning more about resuming a project (steps 8–10) should check out Bennatan’s On Time within Budget: Software Project Management, Practices and Techniques (John Wiley & Sons, 2000) as a companion book. You’ll learn what the rebuilt team needs to do to keep the project progress steady until completion.

Just a final note—the title is a bit misleading. Catastrophe implies that the project suddenly became a calamity, but this isn’t the case: projects become late one day at a time, deliveries are of poor quality one bad review at a time, and relationships deteriorate one unaddressed complaint at a time. Similarly, disentanglement makes readers think that the recovery process is painful and slow. Instead, it’s more appropriate to think in terms of “project tsunamis” and “damage control” because the process described is simple and effective.

Naseem Mariam is an architect and engineering manager at Aricent Chennai. Contact her at naseem.mariam@aricent.com.

Understanding Outsourcing

Kevin C. Desouza

Outsourcing concerns all software professionals—from engineers to managers, from professionals to academics, from Chicago to Mumbai. Outsourcing efforts have moved from simple cost-cutting initiatives to sourcing complex knowledge work. In addition, traditional centers for receiving outsourced work—such as the software hubs of Bangalore, India—are now resourcing work to other global locations.

Having recently coauthored a guidebook for executives involved in outsourcing (The Outsourcing Handbook, Kogan Page, 2006), I was eager to review Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce by Erran Carmel and Paul Tjia.

Carmel and Tjia have accomplished quite a feat by combining blended practitioner experiences with sound academic knowledge. Indeed, their colleagues co-authored many of the chapters.

The fundamentals

Section 1 has four chapters. The first sets the stage by tracing outsourcing’s historical roots, analyzing its progress, and outlining the key advantages and challenges for organizations. One highlight is the discussion of the stage model of offshoring.

Next, the authors present an excellent framework for calculating outsourcing efforts’ true cost. Their coverage of the associated risks—such as loss of proprietary knowledge and risks to intellectual property, data security, and corruption—is comprehensive and accurate. I was a bit disappointed, however, that they only devote a page to discussing strategies that managers can use to mitigate these risks and exploit the advantages.

The book then discusses assessing an organization’s readiness for offshore outsourcing and identifying and evaluating providers. The coverage of critical outsourcing components (such as choosing the right project, developing requests for information and proposals, and even contract negotiations) is rich. However, by jamming a lot of content in here, the authors sacrificed some depth.

Finally, this section discusses the three tiers of software-outsourcing export countries:

- Tier 1: India, Ireland, Russia, China, and Israel;
- Tier 2: Brazil, Mexico, South Korea, and the Philippines; and
- Tier 3: countries that have only recently begun outsourcing.

The authors explain how to choose the right country to source work to by examining the risks (such as political instability and regulatory changes) and incentives from the national government.

Managerial competency

The second section begins with a chapter that’s a must-read for anyone considering offshore outsourcing as a business strategy. Carmel and coauthor Peter Schumacher discuss cost savings, then move beyond this narrow-minded mentality to truly leveraging outsourcing as a business strategy. Offshore outsourcing can increase an organization’s speed, agility, and flexibility, but it’s not a panacea.

In chapter 6, Rebecca Eissner discusses the legal issues surrounding offshore outsourcing, covering hot buttons such as intellectual property protection, labor and employment rights, and data transfer and privacy issues. She also briefly discusses various outsourcing deal and agreement structures and highlights key service concepts in legal agreements such as termination rights, dispute charges, right of approving personnel and subcontractors, and service-level agreements.

Chapter 7, coauthored by Carmel and Erik Beulen, focuses on knowledge management, change management, and governance. They illustrate change-management intricacies via a case study and briefly discuss service-level agreement structures for coordinating outsourcing projects. Overall, I thought this chapter was too superficial; the authors crammed in a lot of topics, each of which easily merits its own chapter.

I really liked chapters 8 and 9, though. Chapter 8 discusses managing distance and time in outsourcing engagements, covering all the essential issues. It’s an excellent read, full of practical, applicable insights. It covers key issues and offers astute guidelines on managing these issues. Chapter 9 is equally good, focusing exclusively on cross-cultural management issues—a wise decision on the authors’ part. They discuss various cultural orientations and explicate why culture matters, commendably walking the fine line of making their point without demeaning any culture. The chapter concludes with practical, sound, and valuable steps to improve cross-cultural communication.

Other stakeholders

The third section discusses building software industries in developing nations, addressing how a country might become a Tier 1 outsourcing provider. They explain how to choose a national strategy for tapping into the outsourcing service arena by examining target strategies: attracting foreign R&D activities or providing generic IT services, foreign R&D activities, or IT-enabled services. The authors also explain why countries should invest in creating a software export industry and the success factors in becoming a global software exporter.

The section also addresses issues that newcomers face—something I would have excluded from the book in favor of covering other issues more adequately. You could find most of this material in an introductory strategic management textbook. Furthermore, building a sound business case and thoroughly analyzing strengths, weaknesses, and threats, aren’t specific to offshore sourcing—you need them to keep any business afloat.

The section concludes by discussing the political issues associated with outsourcing, including backlash and the long-term policy considerations facing countries and organizations.

Offshoring Information Technology is interesting and easy to read. The authors provide vivid vignettes and case studies to drive their arguments home. Carmel and Tjia have put together a well-articulated, well-organized, and timely book on offshore IT outsourcing.

Kevin C. Desouza is an assistant professor at the University of Washington’s Information School. Contact him at kdesouza@u.washington.edu; http://faculty.washington.edu/kdesouza.