Microsoft

BEST PRACTICES

PRACTICAL PROJECT INITIATION

A Handbook with Tools

Karl E. Wiegers

Two-time winner of the *Software Development* Productivity Award

“The wonderful—and scary—thing about Karl's writing is that he reminds you about all of the things that you need to do to be as successful as possible in a project. It is wonderful, because it is nice to have a comprehensive list. It is scary, because when you enumerate all of them, the list seems simply overwhelming, and you might be tempted to therefore not do any. Resist this temptation, scan the list, and then get started answering his questions as best you can. While you may not have all the answers, having a good set of questions is a great way to get started.”

Luke Hohmann
Founder and CEO, Enthiosys, Inc.

“Karl Wiegers's *Practical Project Initiation* offers what its title promises—practical techniques, tips, advice, and tools for one of the most important things a project manager must do well: start projects on a sound path to success. Using a direct and easy-to-read style, Wiegers provides a set of interlocking themes to consider and act upon. He provides pragmatic tools you can implement right away. You will use this book to cut to the chase and find the bottom-line advice you need to get your projects off to a good start.”

Ellen Gottesdiener
EBG Consulting
Author of *Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements* (Goal/QPC, 2005) and *Requirements by Collaboration: Workshops for Defining Needs* (Addison-Wesley, 2002)

“If it's about building software and it's by Karl Wiegers, it's surely worth reading.”

Robert L. Glass
President, Computing Trends

“All project teams can benefit from improvement, but when they want to improve, they need concise information that can be used out of the box. Karl has a long history of providing information that is easy to understand and use out of the box. Whether it is estimation, setting priorities, or negotiation, you can quickly find the relevant section of the book, read it, and put it to use. This book is a great collection of essential, easy-to-understand tools.”

Neil Potter
The Process Group
Co-author of *Making Process Improvement Work* (Addison-Wesley, 2002)

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2007928773

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9   QWT   2 1 0 9 8 7

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Acquisitions Editor: Ben Ryan
Developmental Editor: Devon Musgrave
Project Editor: Victoria Thulman
Editorial Production Services: S4Carlisle Publishing Services

Body Part No. X13-82872

Contents at a Glance

Table of Contents

Preface

Project initiation is the process of formally conceiving, approving, and launching a new project. You don't really have a project until the appropriate stakeholders have approved it during initiation. The time and thought invested during initiation lays the groundwork for all the project work that follows.

All project managers know of certain steps to take at the beginning of a project. They need to develop a business case, write a project charter, obtain sponsorship and funding, assign a project manager, assemble the team, acquire other resources, and develop a project plan. I'm so confident that you already know about these essentials that, with the exception of developing a project charter, I don't address them in this book.

However, numerous other activities are also vital to getting a project off to a good start. Unfortunately, project managers sometimes gloss over these steps. Perhaps they haven't had enough experience to realize how important these steps are. Or maybe they don't feel they can spend the time during the frenzy of project launch. But seasoned managers know that attending to these critical activities can separate success from failure.

This book describes many actions that lay the foundation for a successful project, actions to take during project initiation. Most of the chapters are based on articles I have published previously on various aspects of project management, although several new chapters are included as well. Both experienced and novice project managers will find the practices described here to be valuable. The focus is on software projects that are following any life cycle or methodology (including agile), but the practices apply just as well to nonsoftware projects. As with all process-related guidance, each project team needs to adapt my recommendations to the nature and size of their project. Nonetheless, I believe that *every* software project will benefit from thoughtfully applying the project initiation practices described here.

The book contains 15 chapters grouped into five parts. Part I, "Project Management Fundamentals," contains two chapters that describe some basic concepts and practices of project management in general. Some of these apply just to the project initiation phase, whereas others are applicable throughout the project's duration. A third chapter presents a method and a spreadsheet tool for prioritizing projects in your backlog of requests. There's no point in initiating a project unless you're confident the organization is spending its money on the right project.

The five chapters in Part II, "Preparing for Success," present concrete guidance on how to perform some of the most important activities that will increase the chance of a happy outcome. Three chapters address identifying project stakeholders and their success criteria, determining product release criteria, and identifying and planning to control project risks. Chapter 7 describes the project charter, a key initiation deliverable that makes sure the stakeholders share a common understanding of where the project is headed. Another aspect of software project initiation is to establish the necessary tool infrastructure for your team. Therefore, Chapter 8 presents several lessons I've learned from tool adoption in various organizations.

Project managers always hope for the best, but they should also plan for reality. Part III, "Living with Reality," describes several ways to incorporate a heavy dose of reality into the project, beginning during initiation. Chapter 9 addresses the need for all project participants to negotiate realistically achievable commitments. Chapter 10 describes how to incorporate contingency buffers into project schedules to account for the inevitable surprises, underestimates, and risks that materialize. Finally, Chapter 11 presents the Wideband Delphi group estimation technique, which can yield more accurate estimates than any one individual will generate.

I began my career as a research chemist. As a scientist, I appreciated the importance of data and the insights data can provide. Since I moved into the software domain decades ago I've extended this interest into measuring various aspects of software development activities. Project initiation is the right time to determine what metrics your team needs to provide adequate visibility into progress. It's also the right time to begin growing a measurement culture in your team. The two chapters in Part IV, "Measuring What Happens," address the important topic of software metrics. Chapter 12 presents a concise primer on the concepts and practice of software measurement. Chapter 13 warns about 10 traps that can inhibit the success of measurement programs.

Project initiation is an excellent time to study the lessons garnered from previous projects. Chapter 14 in Part V, "Learning Continuously," suggests ways to establish and use a lessons-learned repository. It also stresses the importance of thoughtfully selecting industry best practices to apply and worst practices to avoid.

When a project team delivers its product the team members experience a range of emotions. They're relieved to be done, they're proud of the achievement, they're tired, and they're ready to move on to a new and interesting activity. But before they move on, the wise project manager will take the time to harvest knowledge from the project that can benefit future work. Therefore, the book concludes with a chapter on how to perform project retrospectives, which are the principal source of those lessons the project manager needs to study during project initiation.

Each chapter begins with a short scenario, typically describing an actual project experience I've had that conveyed a significant message. The scenarios illustrate problems that can arise if a project manager neglects that chapter's principles and practices. A speech balloon icon in the margin highlights other true stories from real projects that support the points made in the chapter. The chapter then presents a tutorial that will let you put the chapter's topics into action. Common project initiation traps to avoid are flagged with an icon of a maze in the margin.

Every chapter ends with several practice activities. Worksheets are included so you can begin to apply the chapter's suggestions to your own project immediately. In addition, the book is accompanied by a number of procedure descriptions, templates, spreadsheet tools, and the like. You may download these process assets from the companion Web site for this book: www.microsoft.com/mspress/companion/9780735625211. (To take advantage of these tools, you will need to be running Word, Excel, and Adobe Acrobat.) The worksheets from the end of each chapter are also available at that Web site so that you can make additional copies for use on other projects.

In my view, there is really no such thing as project management. What we call "project management" is a composite of managing many other project elements: people, requirements, commitments, resources, change, risks, opportunities, expectations, technology, suppliers, and conflicts. Nearly every project includes these elements and the successful project manager must keep an eye on them all. Use the guidance provided in this book to help you launch your next project on a solid foundation.

Acknowledgments

I appreciate the many helpful review comments provided by Jim Brosseau, Patt Dronen, Chris Fahlbusch, Ellen Gottesdiener, Andre Gous, Michael Green, Karen King, Lina Kurtysh, Linda Lewis, Jeannine McConnell, Ulla Merz, Neil Potter, Kathy Rhode, Jean Richardson, Johanna Rothman, Pam Rupert, Mary Sakry, Maureen Stankay, Donna Swaim, and Scott Whitmire. I'm grateful to Dan Read for suggesting that I write this book.

Thanks also go to the editing and production team at Microsoft Press for another fine job: Ben Ryan, Devon Musgrave, Victoria Thulman, and S4Carlisle Publishing Services.

Finally, as always, thanks to my wife, Chris Zambito, for her patience as I spent yet more hours at the keyboard. No surprises this time!

Chapter 6

Know Your Enemy: Introduction to Risk Management[1]

In this chapter:

When my wife and I moved from Rochester, New York, to Portland, Oregon, we had a great project plan. We had a realistic schedule for daily travel distances, routes and hotels selected, and sightseeing activities planned along the way. None of our planning, however, considered the possibility of hitting the world's largest pothole in the middle of South Dakota, which cracked a wheel rim and caused a slow air leak from that tire. Nor did we anticipate the major wreck we had in Boise, Idaho. These unforeseen events were low-probability risks with high impacts. We eventually made it to Portland, but sometimes I think Lewis and Clark had an easier trip west.

Software engineers are eternal optimists. When planning software projects, we usually assume that everything will go exactly as planned. Or, we take the other extreme position: the creative nature of software development means we can never predict what's going to happen, so what's the point of making detailed plans? Both of these perspectives can lead to software surprises, when unexpected things happen that throw the project off track. In my experience, software surprises are never good news.

Risk management has become recognized as a best practice in the software industry for reducing the surprise factor (Brown 1996; DeMarco and Lister 2003). Although we can never predict the future with certainty, we can apply structured risk management practices to peek over the horizon at the traps that might be looming. Then we can take actions to minimize the

1 This chapter was originally published in *Software Development,* 1998, 6(10): 38–42. It is reprinted here, with modifications, with permission of CMP Media Inc.

likelihood or impact of these potential problems. Risk management means dealing with a concern before it becomes a crisis. This improves the chance of successful project completion and reduces the consequences of those risks that cannot be avoided.

During project initiation take the time to do a first cut at identifying significant risks. At this stage it's most important to consider business risks, the risks of either undertaking or not undertaking the project. The project charter template described in Chapter 7 includes a slot for business risks. It's possible that the risks will outweigh the potential benefits of the project. More likely, getting an early glimpse of potential pitfalls will help you make more sensible projections of what it will take to execute this project successfully. Build time for risk identification and risk management planning into the early stages of your project. You'll find that the time you spend assessing and controlling risks will be repaid many times over.

What Is Risk?

A "risk" is a problem that could cause some loss or threaten the success of your project, but which hasn't happened yet. And you'd like to keep it that way. These potential problems might have an adverse impact on the cost, schedule, or technical success of the project, the quality of your products, or team morale. Risk management is the process of identifying, addressing, and controlling these potential problems before they do any harm.

Whether we tackle them head-on or keep our heads in the sand, risks have a potentially huge impact on many aspects of our project. The tacit assumption that nothing unexpected will derail the project is simply not realistic. Estimates should incorporate our best judgment about the potentially scary things that could happen on each project, and managers need to respect the assessments we make. Risk management is about discarding the rose-colored glasses and confronting the very real potential of undesirable events conspiring to throw the project off track.

Why Manage Risks Formally?

A formal risk management process provides multiple benefits to both the project team and the development organization as a whole. First, it gives us a structured mechanism to provide visibility into threats to project success. By considering the potential impact of each risk item, we can focus on controlling the most severe risks first. We can marry risk assessment with project estimation to quantify possible schedule slippage if certain risks materialize into problems. This approach helps the project manager generate sensible contingency buffers. Sharing what does and does not work to control risks across multiple projects helps the team avoid repeating the mistakes of the past. Without a formal approach, we cannot ensure that our risk management actions will be initiated in a timely fashion, completed as planned, and effective.

Controlling risks has a cost. We must balance this cost against the potential loss we could incur if we don't address the risk and it does indeed bite us. Suppose we're concerned about the ability of a subcontractor to deliver an essential component on time. We could engage multiple subcontractors to increase the chance that at least one will come through on schedule. That's an expensive remedy for a problem that might not even exist. Is it worth it? It depends on the downside we incur if indeed the subcontractor dependency causes the project to miss its planned ship date. Only you can decide for each individual situation.

Typical Software Risks

The list of evil things that can befall a software project is depressingly long. The enlightened project manager will acquire lists of these risk categories to help the team uncover as many concerns as possible early in the planning process. Possible risks to consider can come from group brainstorming activities or from a risk factor chart accumulated from previous projects. In one of my groups, individual team members came up with descriptions of the risks they perceived, which I edited together and we then reviewed as a team.

The Software Engineering Institute has assembled a taxonomy of hierarchically-organized risks in 13 major categories, with some 200 thought-provoking questions to help you spot the risks facing your project (Carr et al. 1993). Steve McConnell's *Rapid Development* (1996) also contains excellent resource material on risk management and an extensive list of common schedule risks.

Following are several typical risk categories and some specific risks that might threaten your project. Have any of these things have happened to you? If so, add them to your master risk checklist to remind future project managers to consider if it could happen to them, too. There are no magic solutions to any of these risk factors. We need to rely on past experience and a strong knowledge of software engineering and management practices to control those risks that concern us the most.

Expecting the project manager to identify all the relevant risks. Different project participants will think of different possible risks. Risk identification should be a team effort.

Dependencies

Some risks arise because of dependencies our project has on outside agencies or factors. We cannot usually control these external dependencies. Mitigation strategies could involve contingency plans to acquire a necessary component from a second source, or working with the source of the dependency to maintain good visibility into status and detect any looming problems. Following are some typical dependency-related risk factors:

- Customer-furnished items or information
- Internal and external subcontractor or supplier relationships
- Intercomponent or intergroup dependencies
- Availability of trained and experienced people
- Reuse from one project to the next

Requirements Issues

Many projects face uncertainty and turmoil around the product's requirements. Some uncertainty is tolerable in the early stages, but the threat increases if such issues remain unresolved as the project progresses. If we don't control requirements-related risks we might build the wrong product or build the right product badly. Either outcome results in unpleasant surprises and unhappy customers. Watch out for these risk factors:

- Lack of a clear product vision
- Lack of agreement on product requirements
- Inadequate customer involvement in the requirements process
- Unprioritized requirements
- New market with uncertain needs
- Rapidly changing requirements
- Ineffective requirements change management process
- Inadequate impact analysis of requirements changes

Management Issues

Although management shortcomings affect many projects, don't be surprised if your risk management plan doesn't list too many of these. The project manager often leads the risk identification effort, and most people don't wish to air their own weaknesses (assuming they even recognize them) in public. Nonetheless, issues like those listed here can make it harder for projects to succeed. If you don't confront such touchy issues, don't be surprised if they bite you at some point. Defined project tracking processes and clear project roles and responsibilities can address some of these conditions.

- Inadequate planning and task identification
- Inadequate visibility into project status
- Unclear project ownership and decision making
- Unrealistic commitments made, sometimes for the wrong reasons
- Managers or customers with unrealistic expectations
- Staff personality conflicts

Lack of Knowledge

Software technologies change rapidly and it can be difficult to find suitably skilled staff. As a result, our project teams might lack the skills we need. The key is to recognize the risk areas early enough so we can take appropriate preventive actions, such as obtaining training, hiring consultants, and bringing the right people together on the project team. Consider whether the following factors apply to your team:

- Lack of training
- Inadequate understanding of methods, tools, and techniques
- Insufficient application domain experience
- New technologies or development methods
- Ineffective, poorly documented, or ignored processes
- Technical approaches that might not work

Outsourcing

Outsourcing development work to another organization, possibly in another country, poses a whole new set of risks. Some of these are attributable to the acquiring organization, others to the supplier, and still others are mutual risks. If you are outsourcing part of your project work, watch out for the following risks:

- Acquirer's requirements are vague, ambiguous, incorrect, or incomplete.
- Acquirer does not provide complete and rapid answers to supplier's questions or requests for information.
- Supplier lacks appropriate software development and management processes.
- Supplier does not deliver components of acceptable quality on contracted schedule.
- Supplier is acquired by another company, has financial difficulties, or goes out of business.
- Supplier makes unachievable promises in order to get the contract.
- Supplier does not provide accurate and timely visibility into actual project status.
- Disputes arise about scope boundaries based on the contract.
- Import/export laws or restrictions pose a problem.
- Limitations in communications, materials shipping, or travel slow the project down.

Risk Management Components

Risk management is the application of appropriate tools and procedures to contain risk within acceptable limits. As with other project activities, begin risk management by developing a plan. The work aids that accompany this book include a risk management plan template; an outline of this template appears in Figure 6-1. The template document includes guidance that describes how to complete each section. This template is suitable for larger projects. Small projects can include a concise risk management plan as a section within the overall project management plan.

1. Purpose
2. Roles and Responsibilities
3. Risk Documentation
4. Activities
5. Schedule for Risk Management Activities
6. Risk Management Budget
7. Risk Management Tools

Appendix. Sample Risk Documentation Form

Figure 6-1 Risk management plan template.

Risk management consists of the subactivities illustrated in Figure 6-2 and described in the next section (Boehm 1989).

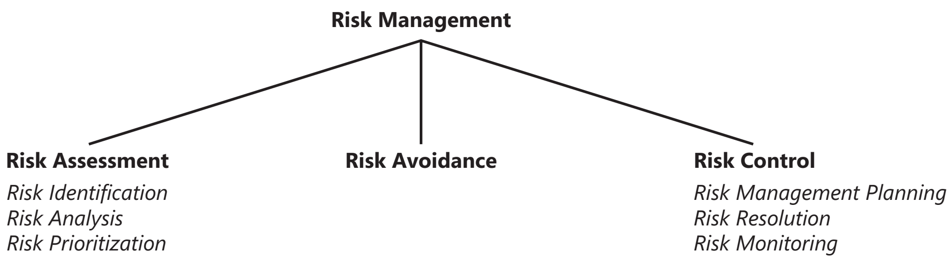

Figure 6-2 Components of risk management.

Risk Assessment

Risk assessment is the process of examining a project to identify areas of potential risk. *Risk identification* can be facilitated with the help of a checklist of common risk areas for software projects, such as the brief lists presented in this chapter. You might also study an organization-wide compilation of previously identified risks and mitigation strategies, both successful and unsuccessful. *Risk analysis* examines how project outcomes might change as a result of the identified risks.

Risk prioritization helps the project focus on its most severe risks by assessing the risk exposure. Exposure is the product of the probability of incurring a loss due to the risk and the potential magnitude of that loss. I usually estimate the probability from 0.1 (highly unlikely) to 1.0 (certain to happen), and the loss (also called impact) on a relative scale of 1 (no problem) to 10 (deep tapioca). Multiplying these factors together provides an estimate of the risk exposure due to each item, which can run from 0.1 (don't give it another thought) through 10 (stand back, here it comes!). It's simpler to estimate both probability and loss as High, Medium, or Low. Table 6-1 shows how you can estimate the risk exposure level as High, Medium, or Low by combining the probability and loss estimates. It's also a good idea to consider the time horizon during which a risk might pose a threat. Confront imminent risks more aggressively than those for which you still have some breathing space.

Table 6-1 Estimating Risk Exposure from Probability and Loss

	Loss		
Probability	**Low**	**Medium**	**High**
Low	*Low*	*Low*	*Medium*
Medium	*Low*	*Medium*	*High*
High	*Medium*	*High*	*High*

Risk Avoidance

Risk avoidance is one way to deal with a risk: don't do the risky thing! You might avoid risks by not undertaking certain projects, or by relying on proven rather than cutting-edge technologies when possible. In certain situations you might be able to transfer a risk to some other party, such as a subcontractor.

Risk Control

Risk control is the process of managing risks to achieve the desired outcomes. *Risk management planning* produces a plan for dealing with each significant risk, including mitigation approaches, owners, and timelines. *Risk resolution* entails executing the plans for dealing with each risk. Finally, *risk monitoring* involves tracking your progress toward resolving each risk item.

Let's look at an example of risk management planning. Suppose the "project" is to take a hike through a swamp in a nature preserve. You've been warned that the swamp might contain quicksand. So the risk is that we might step in quicksand and be injured or even die. One strategy to mitigate this risk is to reduce the probability of the risk actually becoming a problem. A second option is to consider actions that could reduce the impact of the risk if it does in fact become a problem. So, to reduce the probability of stepping in the quicksand, we might be on the alert, looking for signs of quicksand as we walk, and we might draw a map of the swamp so we can avoid these quicksand areas on future walks. To reduce the impact if someone does step in quicksand, perhaps the members of the tour group should rope themselves together. That way if someone does encounter some quicksand the others could quickly pull him to safety. In that way we reduce the impact of stepping in the quicksand. Although, of course, we still stepped in the quicksand.

Even better, is there some way to prevent the risk from becoming a problem under any circumstances? Maybe we build a boardwalk as we go so we avoid the quicksand. That will slow us down and it will cost some money. But, we don't have to worry about quicksand any more. The very best strategy is to eliminate the root cause of the risk entirely. Perhaps we should drain the swamp, but then it wouldn't be a very interesting nature walk. By taking too aggressive a risk approach, you can eliminate the factors that make a project attractive in the first place.

Documenting Risks

Simply identifying the risks facing a project is not enough. We need to write them down in a way that lets us communicate the nature and status of risks throughout the affected stakeholder community over the duration of the project. Figure 6-3 shows a form I've found to be convenient for documenting risks. It's a good idea to keep the risk list itself separate from the risk management plan, as you'll be updating the risk list frequently throughout the project.

ID: <*sequence number or a more meaningful label*>		
Description: <*List each major risk facing the project. Describe each risk in the form "condition-consequence."*>		
Probability: <*What's the likelihood of this risk becoming a problem?*>	**Loss:** <*What's the damage if the risk does become a problem?*>	**Exposure:** <*Multiply Probability times Loss.*>
First Indicator: <*Describe the earliest indicator or trigger condition that might indicate that the risk is turning into a problem.*>		
Mitigation Approaches: <*State one or more approaches to control, avoid, minimize, or otherwise mitigate the risk.*>		
Owner: <*Assign each risk mitigation action to an individual for resolution.*>	**Date Due:** <*State a date by which the mitigation approach is to be completed.*>	

Figure 6-3 A risk documentation form.

Use a *condition-consequence* format when documenting risk statements. That is, state the risk situation (the condition) that you are concerned about, followed by at least one potential adverse outcome (the consequence) if that risk should turn into a problem. Often, people suggesting risks state only the condition—"The customers don't agree on the product requirements"—or the consequence—"We can only satisfy one of our major customers." Pull those together into the condition-consequence structure: "The customers don't agree on the pr oduct requirements, so we'll only be able to satisfy one of our major customers." This statement doesn't describe a certain future, just a possible outcome that could harm the project if the condition isn't addressed.

Keep the items that have high risk exposures at the top of your priority list. You can't address every risk item, so use this prioritization mechanism to learn where to focus your risk control energy. Set goals for determining when each risk item has been satisfactorily controlled. Your mitigation strategies for some items may focus on reducing the probability, whereas the approach for other risks could emphasize reducing the potential loss or impact.

The cell in the form labeled Mitigation Approaches allows you to identify the actions you intend to take to keep the risk item under control. With any luck, some of your mitigation approaches will attack multiple risk factors. For example, one group with which I worked identified several risks related to failures of components of their Web delivery infrastructure (servers, firewall, e-mail interface, and so forth). A mitigation strategy that addressed several of those risks was to implement an automated monitoring system that could check the status of the servers and communication functions periodically and alert the team to any failures.

Figure 6-4 illustrates an alternative template for your risk list, which is also included in the process assets on the Web site that accompanies this book. This format includes essentially the same information that's in Figure 6-3 but it is laid out in a way that is amenable to storing in a spreadsheet or a table in a word-processing document. Storing the risks in a table or spreadsheet facilitates sorting the risk list by descending risk exposure.

Risk Tracking

As with other project management activities, you need to get into a rhythm of periodic monitoring. You may wish to appoint a risk manager or "risk czar" for the project. The risk manager is responsible for staying on top of the things that could go wrong, just as the project manager is staying on top of the activities leading to project completion. One project team dubbed their risk manager "Eeyore" after the Winnie-the-Pooh character who always bemoaned how bad things could become. It's a good idea to have someone other than the project manager serve as the risk manager. The project manager is focused on what he has to do to make a project succeed. The risk manager, in contrast, is identifying factors that might prevent the project from succeeding. In other words, the risk manager is looking for the black cloud around the silver lining that the project manager sees. Asking the same person to take these two opposing views of the project can lead to cognitive dissonance; in an extreme case, his brain can explode!

Keep the top 10 risks highly visible (McConnell 1996) and track the effectiveness of your mitigation approaches regularly. As the initial list of top priority items gradually gets beaten into submission, new risks might float up into the top 10. You can drop a risk off your radar when you conclude that your mitigation approaches have reduced the risk exposure from that item to an acceptable level.

Assuming that a risk is controlled simply because the selected mitigation action has been completed. Controlling a risk might require you to change the risk control strategy if you conclude it is ineffective.

A student in a seminar once asked me, "What should you do if you have the same top five risks week after week?" A static risk list suggests that your risk mitigation actions aren't working. Effective mitigation actions should lower the risk exposure as the probability, the loss, or both decrease over time. If your risk list isn't changing, check to see whether the planned mitigation actions have been carried out and whether they had the desired effect.

Failing to look for new risks that might arise during the course of the project. Conditions can change, assumptions can prove to be wrong, and other factors might lead to risks that weren't apparent or perhaps did not even exist at the beginning of the project.

ID	Description	P	L	E	First Indicator	Mitigation Approach	Owner	Date Due
	<List each major risk facing the project. Describe each risk in the form "condition – consequence". Example: "Subcontractor's staff does not have sufficient technical expertise, so their work is delayed for training and slowed by learning curve.">	**P*	*†L*	*‡E*	*<For each risk, describe the earliest indicator or trigger condition that might indicate that the risk is turning into a problem.>*	*<For each risk, state one or more approaches to avoid, transfer, control, minimize, or otherwise mitigate the risk. Accepting the risk is another option. Risk mitigation approaches should yield demonstrable results, so you can measure whether the risk exposure is changing.>*	*<Assign each risk action to an individual.>*	*<State a date by which each mitigation action is to be completed.>*

Key: **P*= Probability of occurrence of the risk, expressed as a number between 0.1 (highly unlikely) and 1.0 (guaranteed to happen). Alternatively, you could estimate this as Low, Medium, High.

†L= Relative loss if the risk does turn into a problem, expressed as a number between 1 (minimal impact) and 10 (catastrophe). Alternatively, you could estimate this as Low, Medium, High. Even better, estimate the actual loss in terms of calendar weeks for schedule impact, dollars for a cost impact, etc.

‡E= Risk Exposure. If numeric values were assigned to Probability and Loss, then Risk Exposure = P * L. If relative values were used (Low, Medium, High), estimate the overall risk exposure using Table 6-1.

Figure 6-4 An alternative risk list template.

Risk Management Can Be Your Friend

The skillful project manager will use risk management to raise awareness of conditions that could cause the project to go down the tubes. Consider a project that begins with a fuzzy product vision and no customer involvement. The astute project manager will spot this situation as posing potential risks and will document them in the risk list. Early in the project's life, the impact of this situation might not be too severe. However, if time passes and the lack of product vision and customer involvement are not improved, the risk exposure will steadily rise.

By reviewing the risk list periodically, the project manager can adjust the estimated probability and/or impact of these risks. The project manager can escalate risks that aren't being controlled to the attention of senior managers or other stakeholders. They can then either stimulate corrective actions or else make a conscious business decision to proceed in spite of the risks. In this way we're keeping our eyes open and making informed decisions, even if we can't control every threat the project faces.

Learning from the Past

We can't predict exactly which of the many threats to our projects might come to pass. However, most of us can do a better job of learning from previous experiences to avoid the same pain and suffering on future projects. As you begin to implement risk management approaches, record your actions and results for future reference. Try these suggestions:

- Record the results of even informal risk assessments, to capture the thinking of the project participants.
- Document the mitigation strategies attempted for each risk you chose to confront, noting which approaches worked well and which didn't pay off.
- Conduct retrospectives to identify the unanticipated problems that arose (see Chapter 15). Should you have been able to see them coming through better risk management, or would you likely have been blindsided in any case? Could these same problems occur on other projects? If so, add them to your growing checklist of potential risk factors for the next project to consider.

Anything you can do to improve your ability to avoid or minimize problems on future projects will improve your company's business success. Risk management can also reduce the chaos, frustration, and constant firefighting that impair the quality of work life in so many software organizations. The risks are out there. Find them before they find you.

Practice Activities

1. On Worksheet 6-1 list some risk categories that might threaten the success of your project. Identify several specific risk factors in each category.
2. Use Worksheet 6-2 to document several of the risks you identified in the previous practice activity. Be sure to describe each risk in the form of a condition followed by one or more possible consequences. Plan how you might control each risk.

Worksheet 6-1: Your Risk List

Risk Category	Specific Risks

Worksheet 6-2: Documenting Risks

ID	Description	P	L	E	First Indicator	Mitigation Approach	Owner	Date Due
1								
2								
3								
4								
5								
6								

Chapter 9

Negotiating Achievable Commitments[1]

In this chapter:

I once observed a discussion between Steve, a senior manager, and Rob, a project manager, about a new project. "How long will this project take?" Steve asked. "Two years," replied Rob. "No," said Steve, "that's too long. I need it in six months." So what did Rob say? "Okay." Now, what changed in those few seconds? Nothing! The project didn't shrink by a factor of four. The development team didn't get four times larger (although this would not have solved the problem). The team didn't become 400% as productive. The project manager simply said what he knew Steve wanted to hear. Not surprisingly, the project took more than two years to complete.

Too many software projects are launched with the hope that productivity magic will happen, despite previous experiences to the contrary. Sometimes a team does get lucky or pulls off a miracle. More often, though, miracles don't happen, leaving managers, customers, and developers frustrated and disappointed. Successful projects are based on realistic commitments, not fantasies and empty promises. Project initiation is the right time to establish a convention for how you're going to handle commitments on your project. This chapter presents several ways to improve your ability to make—and keep—achievable project commitments (Humphrey 1997).

Expecting people to take imposed commitments seriously. The commitments you strive hardest to satisfy are those you make freely, not under duress.

1 This article was originally published in *The Rational Edge*, January 2002. It is reprinted here, with modifications, with permission from IBM.

Making Project Commitments

A commitment is a pact one person makes with another. Both parties expect the pact to be kept. The commitment might involve delivering a specific work product or performing a particular service by a specified time and at a certain level of quality. A software development project involves many commitments between customers, business managers, development managers, and individual team members. The project manager makes both individual commitments and commitments on behalf of the whole team. The project manager also requests commitments from team members and perhaps from external stakeholders. Some software projects have at least one formal commitment: a legally binding contract.

Effective commitments must meet two conditions:

1. *We must make them freely.* Powerful people can pressure us to say yes, but most people are unlikely to take ownership of an impossible commitment that is thrust upon them. Some people will pretend to make a commitment they don't intend to fulfill to bring an unpleasant conversation to a close, but this undermines the commitment ethic.
2. *They must be stated explicitly and be clearly understood by all parties involved.* Ambiguity about the specifics of the commitment reduces the chance that it will be satisfied.

It's easy to make unrealistic promises on software projects. Most developers are optimistic about their talents and overestimate their productivity. Project managers assume their staff have 40 or more hours of productive project time available each week, although the reality is considerably less. Managers and customers who don't understand software development pressure the team to provide perfect estimates and to meet their aggressive business expectations with little regard for the uncertainties we face. Often, we get incomplete or misleading information about the problem we're asked to solve, which limits our ability to make meaningful estimates and commitments. Requirements are volatile, changing frequently for both good and not-so-good reasons. We need the freedom to renegotiate unrealistic commitments that turn out to be based on inaccurate information or premises that have changed.

Negotiating Commitments

Negotiations are in order whenever there's a gap between the schedule or functionality project stakeholders demand and your best prediction of the future as embodied in project estimates. Plan to engage in good-faith negotiations with customers, senior managers, and project team members about achievable commitments. *Principled negotiation*, a method to strive for mutually acceptable agreements, involves four key precepts (Fisher, Ury, and Patton 1992):

- Separate the people from the problem
- Focus on interests, not positions
- Invent options for mutual gain
- Insist on using objective criteria

Separate the People from the Problem Negotiation is difficult when emotions get in the way. It's also hard to negotiate with individuals who wield more organizational power than you do. Before you dismiss a manager or customer as an unreasonable slave driver who makes impossible demands, recognize his legitimate concerns and objectives, including commitments others might have made that put *him* in a bind.

Focus on Interests, Not Positions A marketing manager who establishes a strong position in a negotiation ("This product absolutely must ship by November 5th!") can find it difficult to change that position. If you, the project manager, define an opposing but equally entrenched position ("We can't possibly be done before mid-January!"), conflict is inevitable. Rather than reaching an impasse by defending your terrain to the death, seek to understand the interests behind each party's stated position.

Perhaps marketing's real interest is to have a demo version available for an important trade show. Your interest might be to avoid burning out your team with massive overtime for four months and establishing a track record of failure. Maybe you and the marketing manager can agree on a core set of features and performance targets that would satisfy marketing's primary interest *and* be achievable by the convention deadline. Delivery of the completed product could then be deferred to a more reasonable date.

Invent Options for Mutual Gain Negotiation is the way to close the gap between polarized positions. Begin with a win-win objective in mind (Boehm and Ross 1989), and look for creative ways to satisfy each party's interests. Think of feasible outcomes to which you are willing to commit and present those as options.

If you can't commit to fully satisfying a key stakeholder's demands, state what you *are* able to deliver. Could you make a more ambitious commitment if certain conditions were met, such as shedding your team's maintenance responsibilities for existing products? If your team is pressured to work a lot of unpaid overtime to get the job done, will the company grant them compensating vacation time afterward?

Insist on Using Objective Criteria Negotiations based on data and analysis are more constructive than those based on faith and opinions. Data from previous projects and estimates based on a rational planning process will help you negotiate with someone who insists his grandmother could finish the project in half the time you've proposed. When you're relying on commitments from a third party, such as a subcontractor or a vendor, trust data and history over promises. Keep these commitment tips in mind:

- When requesting commitments from other people, remember that an estimate is not the same as a promise. Expect to receive a range for an estimate, not a single number or date. Alternatively, request a confidence level, a probability that the estimate will be met.
- A common reason for commitment failure is making "best case" commitments rather than "expected case" commitments. Some teams define internal *target* delivery dates that are more optimistic than their publicly *committed* delivery dates (see Chapter 10, "Saving

for a Rainy Day"). This sensible approach helps compensate for imperfect estimates and unanticipated eventualities.

- Improve your ability to meet commitments by creating more realistic estimates. Base your estimates on data of actual performance collected from previous activities.
- To avoid overlooking necessary work, use planning checklists that list all the activities you might have to perform for common tasks.
- Contingency buffers provide protection against estimation errors, erroneous assumptions, potential risks that materialize, and scope growth (see Chapter 10).

Once some project requirements are defined, one approach to making commitments is to have team members "sign up" for specific task responsibilities (McConnell 1996; Jeffries, Anderson, and Hendrickson 2001). This is a characteristic of an organization with a healthy commitment ethic. Such voluntary personal commitments—based on each person's evaluation of what it will take to accomplish a goal—motivate team members to make the project succeed. A manager cannot impose this level of enthusiasm and commitment upon others.

Your committed delivery dates might be farther out than your managers or customers would like, but if you consistently fulfill the commitments you make, other people will know they can count on you and your team.

Documenting Commitments

When my brother Bruce was an engineering manager, periodically he would discuss expectations with certain individuals on his team. Sometimes Bruce would think a team member had made a commitment, but the team member viewed Bruce's request merely as a suggestion. To reduce ambiguity, it's helpful to write a brief summary of each commitment that you exchange with someone else. This confirms the communication and establishes a shared expectation of accountability. A written commitment record also permits tracking actual performance against expectations. Unless I write them down, I find it easy to lose sight of promises I've made to others and of things I'm expecting to receive from other people. A simple table such as that in Figure 9-1 is a convenient way to document commitments.

Verbal commitments that are unclear and undocumented. Negotiate to make interpersonal and intergroup commitments explicit and realistic. Renegotiate promptly if changes in circumstances won't permit you to keep them.

Commitment	Made By	Made To	Date Due	Comments

Figure 9-1 Template for a commitment record.

Modifying Commitments

Many projects have formal agreements to implement a specific set of requirements by a particular date. Such commitments can fall by the wayside if the requirements turn out to be technically unfeasible or especially challenging, if customers modify them, or if the stated requirements were just the tip of the *real* requirements iceberg. Project stakeholders must alter their expectations and commitments in the face of evolving information.

Inevitably, project realities—requirements, resources, technologies, budgets, schedules—will change. Problems will arise, risks will materialize, and new functionality will be requested. If your customers demand 10 new features late in the development cycle, they can't expect to get the product by the original deadline. Whenever it appears that some targeted deliverables won't be completed before you run out of time and resources, the key project stakeholders need to decide how to respond. As described in Chapter 4 ("Success Criteria Breed Success"), you need to remember which project dimensions are constraints, which are important success drivers, and which ones you can adjust if necessary. Here are some options you might explore, guided by your established project priorities:

- Can some lower-priority requirements wait for a later release?
- Can delivery be delayed? By how long?
- Can you channel more developers onto the project?
- Will the customer pay extra to bring contractors on board to implement the new features?

If it becomes apparent that you or your team won't meet a commitment, inform the affected stakeholders promptly. Don't pretend you're on schedule until it's too late to make adjustments. Letting someone know early on that you can't fulfill a commitment builds credibility and respect for your integrity, even if the stakeholders aren't happy that the promise won't be kept.

Rigidly adhering to impossible or obsolete commitments can lead to serious problems. One project manager promised year-end delivery of a mission-critical application. He maintained that his small team could do the job without additional resources, but the year drew to a close without a delivery. The next year, his manager brought more team members on board and the project made good progress. Again, the project manager promised to deliver by the end of that year but again he failed to do so. Unwilling to admit that he couldn't achieve his commitment, he kept the challenges he was facing to himself, hoping to save face.

In fact, his stubbornness had the opposite effect. His development team lost credibility with both customers and managers. In addition, technology changes over the long course of the project rendered the new application largely irrelevant. When the system finally was delivered, it was nearly dead on arrival and many stakeholders were disappointed. A more responsible manager would have acknowledged reality and renegotiated resources and delivery dates. And when it became clear that pursuing the initial requirements amounted to throwing good money after bad, the project's decision makers should have redirected or canceled the project.

Your Personal Commitment Ethic

Well-intentioned people often commit to more than they can handle. I once managed a capable and dedicated developer who always said "yes" whenever I asked her to take on a new responsibility. I soon learned, however, that often she didn't deliver on schedule. Her in-basket was overflowing. There was simply no way she could complete everything to which she had agreed, despite her cooperative nature and good intentions.

The same sort of thing happened with this very book. The original publisher concluded partway through the book production process that he was overextended and backed out of the contract. It could have been worse. The book could have lingered for months in an incomplete state, only to be cancelled after a lot of time and goodwill had passed under the bridge. But the contract cancellation did leave me with the problem of scrambling to find another publisher. As you can see, I was successful!

Accepting more responsibilities than you can fulfill makes you look unresponsive and makes others skeptical about your promises. A meaningful commitment ethic includes the ability to say "no," or at least "whoa" (Karten 1994). Here are some other ways to say "no" that might sound more palatable:

- "Sure, I can do that by Friday. What would you like me to *not* do instead?" (As project manager, you're responsible for making these priority decisions.)
- "We can't get that feature into this release and still ship on schedule. Can it wait until the next release, or would you rather defer some other functionality?" (This is a way to say "not now" instead of "no.")

- “That sounds like something I can do, but I’m afraid it’s not as high on the priority list as my other obligations. Let me suggest someone else who might be able to help you more quickly than I can.”

Unachievable commitments. People won’t try as hard to achieve commitments they know are impossible. They don’t want to set themselves up for failure.

A track record of making realistic performance estimates and satisfying your commitments earns credibility. That credibility improves your future ability to negotiate problematic schedules or other commitment requests. Before you say, “Sure, no problem” to a request, use the commitment analysis checklist in Figure 9-2 to think through carefully whether you should make the commitment.

Status	Commitment Criteria
❏	All parties fully understand what is being committed to.
❏	I have examined my assumptions and any external dependencies that could affect my ability to fulfill the commitment.
❏	I have identified circumstances that could prevent me from keeping the commitment. The likelihood that these circumstances will occur is: ❏ Not very likely ❏ Somewhat likely ❏ Very likely
❏	The commitment is realistic and achievable, based on what I know today.
❏	I have documented the commitment, along with my assumptions and dependencies.
❏	I will notify the other parties if anything affects my ability to deliver on the commitment.

Figure 9-2 Commitment analysis checklist.

Despite occasional pressure to promise the impossible, I practice this personal philosophy: Never make a commitment you *know* you can’t keep. My personal preference is to undercommit and overdeliver. Keeping that resolve is both part of being a professional and a way to be recognized as someone who is reliable. Once I was discussing process improvement plans with my department’s aggressive and intimidating senior manager, four levels above me. Fred was pressuring me to agree to a timetable that my group had concluded was not remotely feasible. When I resisted, he grudgingly moved his objective out several months, but even that goal was pure fantasy. Finally I took a deep breath and said, “Fred, I’m not going to commit to that date.” Fred literally did not know what to say. I don’t think anyone had ever before refused to make a commitment he demanded.

It would be unprofessional to commit to an objective that I knew was unattainable, I explained. I said, “We’re not going to work any less hard if we have more time to do it, and our morale will be higher if we’re not set up for certain failure.” Reluctantly, Fred agreed to a more plausible target date. He didn’t shout at me or hit me or fire me, although in some situations those might be possibilities you need to consider.

As a project manager, don't pressure your team members to commit to requirements or delivery dates they don't believe to be achievable. If you fear someone is getting in over his head, discuss the commitment details, assumptions, the risks of failing to meet the commitment, and other obligations that could get in the way. Stay on top of the promises you and your team members make to others by recording and tracking requirements and milestones. Also monitor the promises others make to you. Create an environment in which your team members can turn to you for help with negotiation and priority adjustments. Unfulfilled promises ultimately lead to unhappy people and projects that don't succeed, so strive to build a realistic commitment ethic in your team.

Practice Activities

1. Use Worksheet 9-1 to list your personal commitments or commitments you've made on behalf of your team. Indicate whether they are realistically achievable, knowing what you know now. Identify your (or your team's) interests with respect to each commitment and the interests of the person to whom you've made the commitment. Are any of these commitments at risk of not being satisfied?
2. Identify commitments that others have made to you. Assess the impact if any are not satisfied.

Worksheet 9-1: Your Project's or Your Personal Commitments

Commitment	Made By	Made To	Date Due	Achievable?	Your Interests	Their Interests

Index

About the Author

Karl E. Wiegers is Principal Consultant with Process Impact, a software process consulting and education company in Portland, Oregon. His interests include project management, requirements engineering, peer reviews, process improvement, risk management, and metrics. Previously, he spent 18 years at Eastman Kodak Company as a research scientist, software developer, software manager, and software process and quality improvement leader. Karl has a Ph.D. in organic chemistry from the University of Illinois.

In addition to *Practical Project Initiation: A Handbook with Tools*, Karl is the author of the books *Software Requirements, 2nd Edition* (Microsoft Press, 2003); *More About Software Requirements* (Microsoft Press, 2006)*; Peer Reviews in Software: A Practical Guide* (Addison-Wesley, 2002)*;* and *Creating a Software Engineering Culture* (Dorset House, 1996). Karl has written more than 170 articles on software development and management, chemistry, and military history. Karl has served on the Editorial Board for *IEEE Software* magazine and as a contributing editor for *Software Development* magazine. He is a frequent speaker at conferences and professional society meetings. When he isn't in front of the keyboard, Karl is usually playing his guitars, drinking some wine, or reading about military history. You can reach Karl at *www.processimpact.com*.