



EVALUATING PROJECT DECISIONS

Case Studies in Software Engineering



Carol L. Hoover
Mel Rosso-Llopart
Gil Taran

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Hoover, Carol L., 1955-

Evaluating project decisions : case studies in software engineering / Carol L. Hoover, Mel Rosso-Llopart, Gil Taran.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-321-54456-8 (pbk. : alk. paper)

1. Software engineering. 2. Software engineering—Case studies. 3. Project management—Case studies. I. Rosso-Llopart, Mel, 1956- II. Taran, Gil, 1972- III. Title.

QA76.758.H66 2010

005.1—dc22

2009030151

Copyright © 2010 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

ISBN-13: 978-0-321-54456-8

ISBN-10: 0-321-54456-0

Text printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.

First printing, October 2009

Preface

Software engineering professionals and students all over the world make decisions every day that impact the outcome of their software projects. Although some people are very successful, others unfortunately do not take the steps necessary to ensure that their decisions are likely to lead to success. For instance, not asking the right questions generally leads to solutions that do not adequately solve the problems being considered. Likewise, the formulation of faulty assumptions is often the root cause of poor decisions. A common challenge for software professionals is the identification and correction of poor decisions before they can cause even bigger problems. This may be because it is much easier to judge software engineering decisions after the fact than to monitor and evaluate them before their full impact is known.

Through our own professional experiences as software engineers and project managers, we observed firsthand the need for systematic approaches to evaluating and making decisions for software engineering. We observed that, in general, decision making for software projects lacks an analytical approach that can be applied systematically and consistently to increase the likelihood of making decisions that lead to successful projects. As educators of graduate students in the Carnegie Mellon University Master of Software Engineering (MSE) Program, we use textbooks in software engineering that discuss issues involving decisions to be made but do not specifically talk about how to evaluate and make these decisions. We found numerous publications on decision making in the fields of general business and project management. Unfortunately, these works do not address the specific decisions that must be made for software development projects or the nuances of the issues related to these decisions.

We have searched for case studies to help students learn about software engineering issues in the context of real-world scenarios. Though the use of simulated case studies is recognized to be an effective teaching tool for computer science and engineering, we found few published cases that address the specific problems faced by software project managers or software developers and discuss the decision-making aspects of software engineering. To help fill the gap between theory and practice, we decided to write a book that would address the need of both practitioners and students of software engineering to understand decision making in the context of real software projects.

The purpose of this book is to help practitioners, managers, students, and educators acquire the knowledge and skills needed to evaluate and make good decisions as well as to recognize bad decisions for software project management and software development. The book focuses on case studies constructed to help readers

comprehend and apply concepts important to forming, implementing, monitoring, and evaluating critical decisions in real software projects. The case studies are based upon real project scenarios in organizations that differ in industry, size, and budget, but the names of the organizations and stakeholders have been changed to keep the real entities confidential. The book does not exhaustively cover every possible software engineering event that might occur. Rather, the case studies focus on issues and decisions that typically make or break today's software projects.

Readers who have hands-on development or management experience with the full life cycle of a software project will benefit the most from this book. We intend for the book to serve as a handbook of professional guidance for managing the decisions that can lead to software project success. Upper-level undergraduate or graduate students who have some undergraduate-level coursework in software engineering or professional software engineering experience will also find the case materials to be a stimulating way to explore decision making in real-world software projects.

Readers of this book can acquire **knowledge** of issues and problems that notoriously contribute to software project failure. Readers will learn about issues and problems that occur frequently, have a high impact on today's software projects, or are of emerging and growing importance. The case studies illustrate important decisions in the mapping between the user's expectations and the provider's software solution. The case study analyses highlight effective questions to ask to better analyze software engineering issues or problems and to determine the best decisions or solutions to make. The book also discusses indicators to assess whether a decision or solution for a particular software project issue or problem is working.

By reading and analyzing the software engineering case studies, you can develop **skills** needed to identify key factors that characterize software project problems. You can learn how to analyze the issues and problems that relate to a specific software project scenario. The case analysis activities will guide you through the study of alternative decisions that could be made to address the case issues. They will help you develop ideas about how to apply the case concepts to real software projects. We hope that you will find the case studies to be an interesting way to learn about critical issues in software engineering and about how to apply decision-making techniques to real-world software engineering projects.

Synopses of Book Chapters

Following is a synopsis of the topics and case studies for each chapter. You will notice that all of the chapter titles include the word *managing*. We use *managing* to emphasize the need for project managers and software engineers proactively to identify issues or problems in a particular management area, to analyze and evaluate their nature and importance, to decide upon actions that need to be taken, to implement decisions by taking action, and to monitor results. We would like to promote the idea that all

software professionals, not just managers, need to manage the decisions that they make on the job. The case studies illustrate decisions made or that need to be made by various software project stakeholders. In some cases, the stakeholders manage the decisions in a way that leads to project success. In other cases, the stakeholders do not manage the decisions they make very well and thereby encounter project problems and sometimes failure.

Chapter 1: Managing Decisions

Chapter 1 discusses decision making in software engineering, presents a model for evaluating decisions, and shows how the model can be applied to the decisions made by stakeholders in the case studies. The chapter illustrates the application of the model through several decision scenarios that involve technical issues.

Chapter 2: Managing Requirements

Chapter 2 overviews the following activities involved in managing requirements: eliciting and stating requirements, verifying and validating requirements, negotiating and contracting requirements, managing requirements change and scope, and validating software deliverables with respect to requirements. The chapter provides an in-depth discussion of issues and decisions for each requirements activity. The chapter emphasizes the idea that managing requirements does not end until the software is retired.

Case studies:

- **The New Account Project at HBC:** This case study highlights the activities and decisions involved in establishing the requirements. In particular, the case illustrates problems that can occur regarding decisions about the selection and involvement of stakeholders.
- **On Time, Within Budget, but Wrong:** This case study illustrates how decisions made when establishing and managing requirements can result in software project deliverables that do not satisfy the user stakeholder's expectations even though the project appears to be going well. In particular, the case scenario covers both functional and nonfunctional requirements including statements about required quality attributes.

Chapter 3: Managing Estimates

Chapter 3 illuminates the importance of estimation in planning and managing software projects and other software engineering efforts. The chapter discusses the following activities involved in managing estimates: understanding requirements,

conceptually designing a solution, performing element-wise decomposition (divide), allocating resources from the bottom up (conquer), allocating overhead, estimating construction, and estimating change management. The chapter presents ways to factor considerations of scope, time, quality, and cost into decisions about project estimates.

Case studies:

- **Estimation as a Tool:** This case study examines process improvement in managing estimates via collecting, studying, and applying historical data. In the case, a software project manager must determine the appropriate inputs to make a good decision regarding an estimate for a job.
- **When a Team Runs a Race:** This case study examines factors that influence estimation and how these factors can lead to poor decisions about estimates. The case also looks at issues related to the concurrent execution of multiple projects with interdependent resources, personnel, and developments.

Chapter 4: Managing Plans

Chapter 4 examines the role of decision making in planning software projects. The chapter overviews planning activities such as the following: defining project objectives, policies, and scope; planning tasks and milestones; planning schedules; planning budgets; staffing and other resource planning; tracking and controlling the budget and schedule; managing midstream changes to project plans; managing processes with respect to project objectives, policies, and other project plans; and managing software development with respect to project objectives, policies, schedule, and budget.

Case studies:

- **To Replan or Not to Replan?:** This case study discusses the evaluation and decision of when, why, and how to adjust the budget, schedule, or staffing plans.
- **Managing Plans Is in the Details:** This case study looks at problems and decisions that a software development organization faces in planning and tracking software projects, especially those that share resources with other projects or engineering groups.

Chapter 5: Managing Product

Chapter 5 overviews the issues and decisions involved in managing software product. The chapter discusses the following activities involved in managing product: proper product definition, development process management, quality assurance and control, configuration management, product delivery and installation, training, field service, and the application of the decision model to make product decisions.

Case studies:

- **New Technology—Is It Always the Best?:** In this case study, the project manager must make a decision regarding the adoption of new development technologies and techniques to create a product that is easier to maintain and less expensive to use.
- **Why Is This Product Wrong?:** In this case study, a software developer is faced with the dilemma that although it seems as though he did everything correctly, he still got the wrong answer.

Chapter 6: Managing Process

Chapter 6 emphasizes the importance of managing the processes for software development and software project management. The chapter discusses activities involved in managing process: defining, selecting, and understanding process; teaching process; measuring process; evaluating process performance; changing the process to improve its effectiveness; and using the decision model to make process decisions.

Case studies:

- **Bank on the Verge:** In this case study, a project manager asks two consultants to evaluate whether his team has made good decisions on process and whether there are things that the team can do better.
- **Damn the Process, Full Speed Ahead:** In this case study, a project manager is asked to review another manager's project that is having problems. The troubled project is behind schedule, and no one has been able to clearly identify what is causing the schedule slippage or what should be done to fix the project's processes. The manager must decide how he can help the troubled project.

Chapter 7: Managing Risk

Chapter 7 emphasizes the importance of managing risk across the project life cycle. The chapter discusses activities involved in managing risk: defining thresholds of success for the project, identifying project risks, formulating statements about risk, communicating information about risk to project stakeholders, mitigating risk, and conducting resource trade-offs in making decisions about how to manage risk.

Case studies:

- **SEWeb and Russoft Technologies:** This case study examines the decisions and risks faced by an academic institution in its attempt to use an offshore development firm, a Russian company located in Moscow, to develop a Web-based system. In particular, the case study exemplifies what can happen when the project stakeholders do only minimal risk management throughout the project.

- **Falcon Edutainment and the RiskSim Project:** This case study focuses on software project risks associated with client-developer interactions, communications, requirements elicitation, and the related decisions.

Chapter 8: Managing People Interactions

Chapter 8 discusses factors that influence interactions between people and shows how an understanding of these can help software professionals to manage their interactions with people on the job. The chapter specifically describes activities involved in managing project stakeholder interactions: understanding scenarios of interaction, understanding factors that influence interactions, deciding upon interaction objectives, evaluating factors that influence an interaction, deciding how to adjust controllable factors for an interaction, and reflecting on past interactions to improve performance in future interactions.

Case studies:

- **To Be or Not to Be: A Sense of Urgency at TestBridge:** In this case study, tough decisions have to be made with respect to the ability and willingness of specific players within the team to contribute as team members in light of an organization that is changing its focus and business direction.
- **A Friend or Foe at Hanover-Tech:** This case study focuses on the need to manage people interactions at the managerial level while balancing opposing constraints that are financial, strategic, personal, and temporal.

Chapter 9: Managing Stakeholder Expectations

Chapter 9 clarifies the nature of stakeholder expectations. The chapter introduces a model for understanding the mapping between customer expectations and solution provider expectations. The chapter discusses the activities involved in managing interactions between software project stakeholders: communicating with the customer stakeholder, managing multiple dimensions of stakeholder expectation over time, managing product and process to satisfy stakeholder expectations, understanding different types of customers, and managing stakeholder perception.

Case studies:

- **TCP Enhancements at Gigaplex Systems:** In this case study, the software project team encounters unexpected problems related to satisfying the customer's expectations as well as to internal team dynamics and the external relationship with the customer.

- **Tough Sell at Henkel Labs:** This case study presents a difficult situation in which stakeholder expectations need to be managed across a global organization with multiple subsidiaries and people from different cultures who reside in different countries and who sometimes have opposing organizational and business objectives.

Chapter 10: Managing Global Development

Chapter 10 discusses issues and decisions related to managing software engineering efforts that are geographically distributed. The chapter illuminates the challenges of communicating across geographical and time differences, understanding and handling cultural differences, managing distributed projects and software development, managing outsourced software development, managing software quality with distributed or outsourced development, and managing expectations across geographically dispersed stakeholders.

Case studies:

- **Globally Distributed Team: FibreNet Project:** This case study highlights issues of communication, collaboration, and trust when a project team is globally distributed. The software project manager needs to decide how to improve these aspects across the different globally distributed project sites.
- **Managing Global Software Development at FibOptia:** This case study looks at problems specific to global software development in managing process, product quality, and cost. A senior manager is trying to decide how she can help her global software development organization make process and product improvements.

How This Book Is Organized

In Chapter 1, “Managing Decisions,” we discuss the nature and importance of decision making for software engineering. We present a decision model that we developed and have used in practice to assist software professionals in evaluating decisions. This chapter provides context information that will help you more easily identify and think about issues and problems that occur in the case studies. You will therefore benefit from reading this chapter before you read the other chapters in the book. The other chapters discuss decision making in the context of well-known management areas for software projects. These chapters broadly cover issues concerning people, process, and technology in the context of making decisions about them. Because evaluating risk should be an integral part of making decisions, all chapters discuss risk in the

context of making decisions for a particular management area. Chapter 7, “Managing Risk,” more generally covers risk management throughout the life of a software project. In addition, since process concerns how something is done, all of the chapters in some way discuss process issues when describing how software professionals manage a particular aspect of software engineering.

Chapters 2–10 have the same organizational structure. The first section of each chapter outlines the objectives for that chapter. The second section sets the context for making decisions in a particular software management area. The context material describes primary activities that software professionals perform to manage a particular project area. Alternatively, the context for Chapter 10, “Managing Global Development,” explains that the activities for managing global development include those for the other management areas with added challenges, constraints, and opportunities. The chapter context also discusses key ideas, practical guidelines, or what-if questions for making decisions in the relevant management area. The context highlights example decisions with some analyzes that apply the PEAK decision model from Chapter 1. Some of the context information appears in tables to help you rapidly assimilate and reference it when needed. The chapters present concepts in the context of making decisions with respect to factors that influence stakeholder expectations for software projects: scope (requirements), time (schedule), quality, and cost (budget). The chapter context also includes references to publications that provide background in the software management area for interested readers. The authors recommend that you read or peruse the chapter objectives and context before reading the case studies.

The next section of each chapter presents two case studies whose scenarios highlight key issues and decisions related to the chapter topic. The first case study includes a full analysis of the case study issues and decisions with respect to the PEAK decision model. The case studies discuss decisions made by the stakeholders in the case scenarios and pose follow-on decisions that need to be made. Some case studies tell you what the stakeholders decide to do to solve the case problems and include the results of these decisions; others leave the evaluation of solutions for the case problems to the case study analyses. A set of questions to stimulate an analysis of the second case study concludes the case study. The last section summarizes important ideas to remember about the chapter. A list of references for all chapters appears at the back of the book.

We would like to help set your expectations regarding the references in the book. We do not intend for the book to be a survey of research and publication in the various software management areas. We have included some selected references that provide background for readers who are new to software engineering or to topics covered in the case studies or that provide supplemental information for interested readers. Some of the references are purposely dated because they provide the original and best definition of particular concepts or because the concepts discussed in these sources are still highly relevant for software projects of today. The referenced materials include books, conference and workshop proceedings, journal and magazine articles, and

online information. They eclectically cover topics, events, or quotes from the subject areas of business management, civil engineering, communications, computer science, decision sciences, English literature, filmography, management and management sciences, networking, psychology, regulatory policy, software engineering, systems engineering, and others that provide background information for the case studies and example decision scenarios.

Some of our reviewers commented that a particular case study seemed to be relevant for different chapters or software project management areas. As you read the book, you too may recognize that a particular case study involves issues from multiple management areas. We purposely designed the cases to be multidimensional. After analyzing the cases in a specific chapter and management area, you might analyze the case studies in other chapters whose issues and decisions involve the same management area. For instance, all of the case studies entail issues and decisions involving scope, time, quality, cost, risk, stakeholder expectations, communications and other people interactions, or some set of these. As another example, case studies in Chapter 7, "Managing Risk," Chapter 9, "Managing Stakeholder Expectations," and Chapter 10, "Managing Global Development," encompass issues related to global software development. The scenario for one of the cases in Chapter 7 involves development by an offshore team, and one of the cases in Chapter 9 looks at establishing projects to be executed by globally distributed branches of a company. Both cases for Chapter 10 involve globally distributed software development teams.

The reviewers for our book requested that we provide full analyses for all the case studies. We talked with our editor about this and together decided that for space and cost reasons it would not be practical to include this additional information in the book itself. Our reviewers also gave us excellent suggestions for topics that they would like to see discussed in case studies. Therefore, we decided to offer a Web site where you can obtain supplemental information at www.andrew.cmu.edu/user/rosso/. This site will provide you with materials such as additional case studies and analyses as they become available. Depending on our time constraints, we will try as best we can to provide you with analyses for the second cases in the chapter.

We hope that you enjoy your journey through this book and that you will use the book as a guide as you encounter issues and decisions in your study and practice of software engineering.

Chapter 1

Managing Decisions

1.1 Chapter Objectives

This chapter introduces concepts about evaluating decisions for software development projects and other software engineering efforts. This chapter will help you understand the purpose, rationale, and application of a model for evaluating decisions. Succeeding chapters will illustrate how to apply the model to decisions being made to manage various aspects of software projects and to handle problems faced by the stakeholders in case scenarios. Managing decisions involves identifying a problem to be solved, formulating and evaluating alternative solutions to the problem, selecting among the alternative solutions, and executing the decision or implementing the solution. The model discussed in this chapter applies to the following phases: identifying the problem, formulating and evaluating alternative solutions, and selecting among alternative solutions or decision. The case studies in the succeeding chapters focus on these phases but may also include the execution aspect of managing decisions.

1.2 Context

A decision involves passing judgment on an issue under consideration. It is commonly understood to be the act of reaching a conclusion or of making up one's mind. All software projects involve making decisions. Even the act of not making a decision is a decision. For example, if a software project manager chooses to ignore a project member's request for more resources, the manager is making a decision not to act and must deal with the consequences of this noncommittal decision. Increasing a software professional's responsibility increases the number of decisions that the

professional must make. This book will explore decisions made by different software project stakeholders and will shed light on how these decisions can be made.

The importance of managing the way in which project decisions are made is evident by the numerous publications that discuss decision making, particularly in the context of managing projects, of managing project risks, and more specifically of managing projects involving product development. The following references are just a few select examples of current publications in these areas. For more about decision making as an integral part of project management, see Cleland and Ireland (2007), McManus (2004), Pollack-Johnson and Liberatore (2006), and Verine and Trumper (2007). For more about the relationship between decision making and risk, see Chapman and Ward (2002), Dillon and Tinsley (2008), Hussey and Hall (2007), and Warkentin et al. (2009). For product development project decisions, see Barry et al. (2006), Gutierrez et al. (2008), Krishnan and Ulrich (2001), Messerschmitt (2004), and Schmidt et al. (2001). With the rise of global development, there is also an increasing interest in the effect of cultural, geographical, and time differences on how decisions are made within organizations and projects. For a current discussion of these issues, see Bourgault et al. (2008), Brett (2001), Espinosa et al. (2007), Mojtahedi et al. (2008), Shore (2008), and Wang and Liu (2007).

In general, software professionals do not understand how to *systematically* make decisions that result in software projects that are considered to be successful by the project stakeholders. One main problem is that decision makers for software projects often do not state or analyze the inputs and outputs of their decision processes specifically with respect to the needs and expectations of the stakeholders. They may recall that a solution was successful for a particular problem, but then they are surprised when the solution is not successful in solving a similar problem in which the stakeholders have different expectations. This book focuses on the viewpoints of different software project stakeholders to help you refine your decision-making processes so that the resulting solutions are more likely to satisfy stakeholder needs and values.

In general, software project stakeholders make decisions to satisfy their expectations regarding the scope of the project deliverables, the time for the project (schedule), the quality of the project deliverables or product, and the cost (budget). Figure 1-1 shows a model of the expectations that stakeholders have for software projects and the project deliverables.

The model shows that the dimensions of stakeholder expectation (scope, time, quality, and cost) are related. Stakeholders trade different values for scope, time, quality, and cost when establishing the requirements for a software project. Prioritizing their expectations can help stakeholders to make trade-offs more easily and effectively. Different software project stakeholders make different decisions to support their interests. They have different inputs feeding their decision process and different expectations about the outcomes of their decisions. For instance, stakeholders have different levels of tolerance for the risks associated with the decisions that they make.

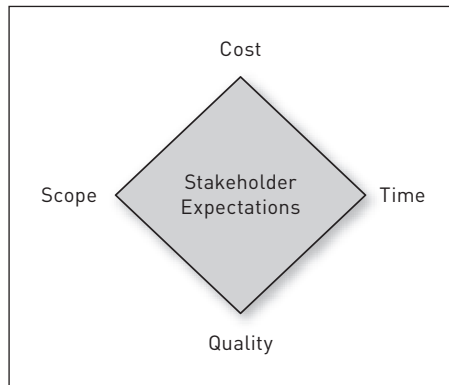


Figure 1–1: *Stakeholder expectation model*

Consider an example of how software stakeholders make decisions based upon their expectations with respect to scope, time, quality, and cost. Suppose a customer decides upon an acceptable budget for a specified work product, schedule, and quality. Likewise, the solution provider establishes a definition of quality to satisfy both the customer’s expectations for the work product as well as the product standards set by the solution provider (who has a reputation in the market to maintain). The solution provider then determines an amount to charge that factors in the cost of developing the work product to satisfy the customer’s expectations regarding scope, time, and quality.

But what happens if the customer’s acceptable cost is lower than the amount that the solution provider wants to charge? This case would generate another set of decisions. Will the customer and solution provider agree to negotiate the amount to be charged? Will the stakeholders consider alternative scopes, schedules, levels of quality, and budgets? Or will the stakeholders decide that they are unwilling to negotiate an amount to be charged that is agreeable to both of them?

The customer may decide to pursue other solution providers and find that the costs posed by these solution providers are significantly higher than the amount asked by the first solution provider. The customer may then discover that the first solution provider in the meantime has taken on other projects and will not be available to perform the work until a future time that is not acceptable to the customer. What will the customer do now? Did something go wrong in the customer’s decision-making process?

This customer–solution provider scenario highlights the interconnected or causal nature of decision making: One good or poor decision frequently leads to another good or poor decision. As the scenario described, a mismatch between the customer’s acceptable cost and the solution provider’s desired price may result in a cascade of successive decisions that eventually may leave the customer with the choice of selecting a solution provider who would charge a higher amount or abandoning the desired work to be done.

Therefore, it is important that stakeholders understand the factors that affect their decisions as well as the potential consequences of their decisions. Project delays and failures are usually related to a series of poor decisions. When asked how a project becomes a year late, Frederick Brooks answered, “One day at a time” (Brooks 1995, 160). We suggest that a more revealing answer is “One decision at a time.”

One way to solve the problem of interconnected decisions is to disconnect them as much as possible, but this solution is not always applicable. For instance, a mismatch between customer and solution provider expectations is common. The issue is how to manage the stakeholder decisions to resolve the mismatch in a way that leads to the best possible outcome. The stakeholders need a systematic way to make software engineering decisions that are likely to lead to successful results. They also need a management strategy to monitor and correct less than optimal decisions before further harm is done to a project.

The purpose of this book is to help you refine your decision-making processes and decision management strategies. We examine the decision-making process with respect to the decision evaluation model presented in the next section. The model is a visualization of factors that are used to make decisions in software development and software project management. The model also provides decision makers with a systematic way to analyze the inputs and outputs of the decision-making process. The book shows how software project stakeholders use their expectations regarding scope, time, quality, and cost as inputs to making project decisions in case scenarios.

The model emphasizes that every decision incurs some amount of assumed risk and that the solution may not succeed in solving the stated problem. It is important for decision makers to understand the risk assumed when making a decision because it may be unacceptable to the project stakeholders. Assumed risk may accumulate over time to a level that is unhealthy for software projects. The objective is to recognize decision situations that need to be carefully managed because (1) the risk assumed in making these decisions is high or (2) the cost associated with unsuccessful outcomes to these decisions is high. The case study analyses in the book show how the model can be applied to make less risky decisions for critical software project management and engineering scenarios. In particular, the case study analyses show how stating the inputs and outputs of the decision model in terms of the stakeholder expectations helps reduce the risks assumed by alternative solutions.

The next section and the remaining chapters in the book will answer questions such as the following within the context of software projects:

- What do decision makers know when making decisions?
- Do decision makers use a process when making decisions?
- Are all decisions of equal value?
- Are software project decisions different from those for other projects? If so, how?
- What does it mean to manage decisions?
- Why is it important to manage decisions?

1.3 Decision Model for Software Engineering

For many people, the decision-making process consists of three basic (and usually vague) stages:

1. Information goes into a decision process.
2. A person-dependent miracle occurs.
3. A solution or decision comes out.

Professionals often consider people to be experts or gurus in their fields if they can make good decisions easily. This section gives an overview of a decision model whose purpose is to help software professionals manage their decision processes so that they better understand the solutions they generate.

The PEAK decision model presented in Figure 1-2 does not indicate “what a particular decision should be.” It simply implies that stakeholders need to carefully examine their decision processes to help ensure that successful decisions are made. When making decisions, stakeholders should consider the inputs and their relationship to the solution as well as the risk assumed with alternative solutions. The model provides insight into how the inputs and outputs of the decision-making process influence the way in which decisions are made.

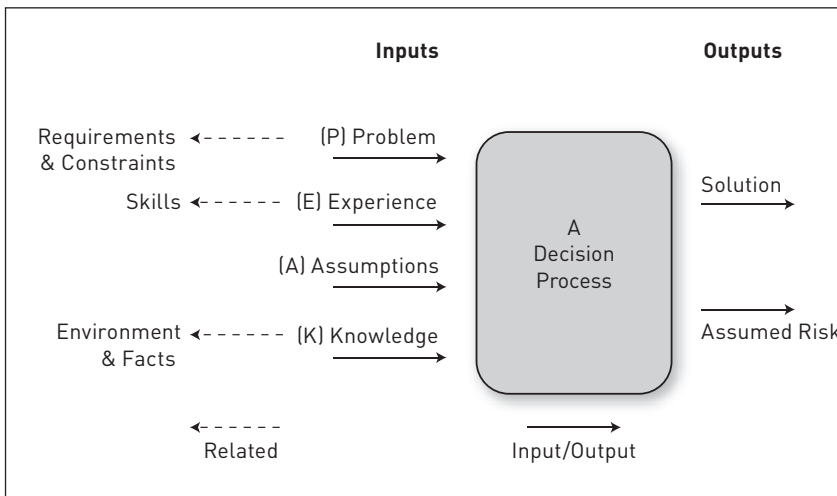


Figure 1-2: PEAK decision model

The *inputs* to the decision model are the following elements:

(P) Problem: What is the issue to be resolved or the problem to be solved? The problem statement should provide a clear representation of what needs to be solved.

(E) Experience: From prior events, what does the decision maker know or know how to do that might help with this problem? Has the decision maker seen or solved a problem like this one before? How appropriate and accurate is the decision maker's historical information?

(A) Assumptions: What information is accepted as fact without having evidence? What information about the problem does the decision maker abstract away because the information is not thought to be relevant to the solution?

(K) Knowledge: What conceptual understanding or factual basis can help the decision maker with the problem? What has the decision maker learned since last dealing with a problem like this? What facts does the decision maker have about the problem? What is the environment that surrounds this problem? For instance, when looking at military problems to be solved, soldiers might ask, "What is the current situation and terrain?"

The *outputs* from the model are the following elements:

Solution: What do the stakeholders need in order to solve the problem or to resolve the issue? What alternative solutions are feasible, and why? What are the benefits and cost associated with each alternative? Have the relevant stakeholders discussed the issues concerning the alternative solutions and expressed their preferences? The stakeholders may not know whether a solution is successful until they implement it. The results, whether successful or not, of implementing a decision feed back into the model as part of the decision maker's knowledge and experience. In response to a colleague who asked whether he thought himself to be a failure, Thomas Edison said, "Not at all. Now, I definitely know more than a thousand ways for how NOT to make a light bulb" (Rovira and Trias de Bes 2004, 1).

Assumed Risk: What is the probability that the solution will not work as envisioned? When a stakeholder makes a decision, the stakeholder assumes some level of risk that the solution or decision will not lead to a successful result. Risk is a consequence of making decisions. What are the risks associated with the alternative solutions? Have the relevant stakeholders discussed these risks with respect to their preferences for the alternatives?

Even if the decision maker is not aware of them, the inputs and outputs of a decision always exist when a decision is being made. To manage their decisions, stakeholders

need to identify and manage these inputs and outputs when making decisions. The case studies in the book will describe situations in which most but not necessarily all of the inputs are known. Each case study will present the problem, the assumptions, and many facts. You will bring other knowledge and experience as inputs to the decision model. This book will help you use the inputs and feasible outputs of the model, as they occur in the case studies, to evaluate your decision processes.

So, why do decision makers need a model for evaluating decisions? A model helps decision makers think about the following aspects of their decision processes:

- Information that should be considered in making decisions
- The impact that the inputs to the decision process have on the outputs
- Solutions that are feasible with respect to the nature of the problem
- The assumed risk associated with alternative solutions
- Ways to improve the management of decisions

The remainder of this section provides four examples to illustrate how you can apply the model to the decision process. These examples clarify how to apply the model to evaluate alternative solutions and then to make a decision by selecting the best alternative. The first example, “Software Test Rerun Problem,” shows how to identify the inputs and outputs to a decision regarding whether to rerun a software test. The second example, “California Bridge Problem,” demonstrates how to apply the model to the more complex problem of building a bridge that can withstand earthquakes, a major issue for the California Department of Transportation. This example demonstrates how newly acquired knowledge feeds back into the decision-making process. The third example, “Unfamiliar Legacy Code Problem,” highlights the risk that is assumed when making a decision. The last example, “Data-Processing Problem,” clarifies why it is important to understand the nature of the real problem before working on a solution. The diagram shown with each story problem summarizes key inputs and outputs for the decision being made.

CASE STUDY

Software Test Rerun Problem

This example involves a testing problem that software developers often face. The scenario for the problem follows. Before leaving the office, Bob started a software regression test on the computer used for testing software products. He expected the test to complete and the

test results to be ready sometime during the night. Bob came into the office the next morning and saw the login prompt on the test computer. He knew either that someone logged out the test account or that the machine rebooted. Bob quickly determined that there had been a power failure and that the machine had rebooted when the power resumed. The test that he had been running did not complete before the power failure, and the complete test results were not available.

Bob examined the incomplete test results and thought they did not provide sufficient verification of the software component being tested that would be needed to integrate this component with other software system components. He reasoned that the incomplete test results probably would not be useful to the software engineers who were depending on them. Bob estimated that running the entire test would take about four hours and that the complete results could be available by the afternoon if he reruns the test this morning. He rationalized that it would be better to have complete test results that are late by half a day than to provide the test stakeholders with incomplete results. Bob decided to rerun the test now because he did not want the delivery of the test results to be a day or more late.

Let's use the PEAK model to evaluate Bob's decision. Figure 1-3 outlines the inputs to and outputs from Bob's decision process. The risk associated with Bob's decision is also an output of his decision-making process.

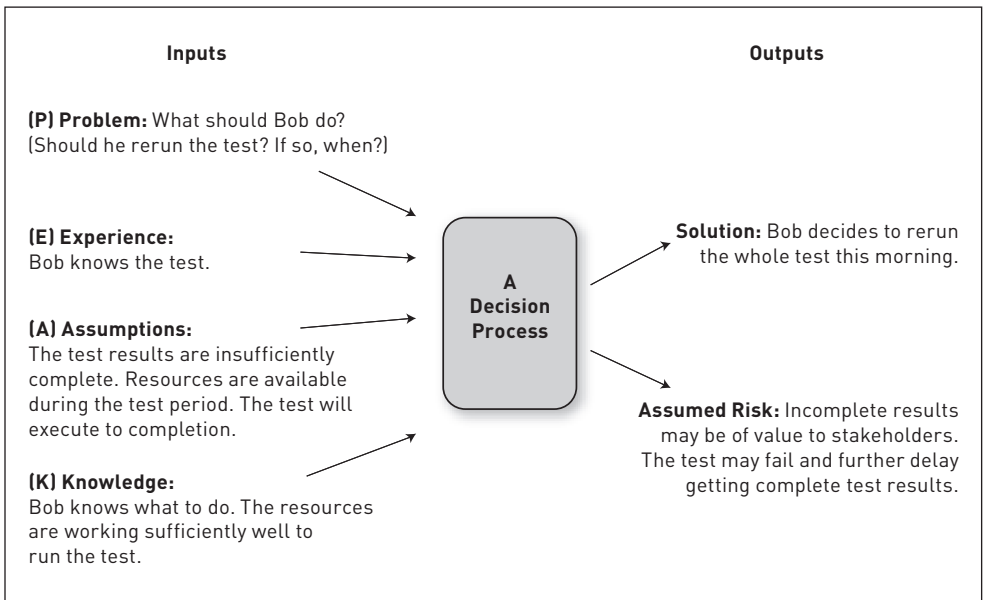


Figure 1-3: Software test rerun problem

Inputs

(P) Problem: The software test failed and yielded incomplete test results. What should Bob do?

(E) Experience: Bob knows the test. (He has run this regression test before. He can estimate the execution time and predict potential causes of failure. Therefore, Bob knows the test can be executed completely by noon if he starts the test now.)

(A) Assumptions:

- The test results are insufficiently complete. (Only complete test results are of sufficient value to the software engineers who will use them.)
- The resources are available during the test period (the next four hours).
- The test will execute to completion without problems. (The problem that occurred last night or any other problem will not occur.)

(K) Knowledge:

- Bob knows what test to run and understands the system well enough to run the test (fact).
- The resources are working sufficiently well to successfully run the test (environment).

Outputs

Solution: Bob did not give the first two alternatives in the following list much consideration and chose the last alternative:

- Do not rerun the test. (Deliver incomplete test results to the software engineers.)
- Rerun the whole test later. (The test results will be a day or more late.)
- Rerun the whole test this morning. (The test results will be approximately half a day late.)

Assumed Risk:

- Incomplete results that are available now may be of value to the software engineers who will use them.
- The test may fail, or the completion of the test results may be delayed for some other reason. (For example, another user may have already planned to use the computer for testing during the day. The execution of another test on the computer may delay or interfere in some way with the execution of the target test. The power outage may occur again.)

You may wonder whether each of the inputs is true. The assumption that “only complete test results are of sufficient value” is particularly suspect. If this assumption is correct, then failure of the test to complete would once again preclude the usefulness of the test results. This constraint would necessitate a more controlled test environment (for example,

a battery backup for the test computer) including more extensive monitoring of the test execution. Bob could have assumed that any results up to the point where the power failure occurred might be valid. His actual assumption may have overly constrained the decision so that he did not consider other alternative solutions, such as delivering partially complete test results and rerunning only the incomplete part of the test or delivering partially complete test results and rerunning the entire test later after verifying that the test computer would be available. In addition, he is assuming that the power failure will not occur again when he reruns the test. What if the test is the cause of the power failure? An unproven assumption, such as the problem will not occur again, introduces risk for a decision based upon this assumption. If one of the assumptions is not true, the decision when acted upon may result in failure. Decision makers reduce the risk associated with assumptions by gathering the information needed to convert them to facts (knowledge inputs).

A critical aspect to understand about the decision model is that the inputs and outputs exist even if the decision maker does not acknowledge them. Bob made the implicit assumption that he has time to rerun the regression test this morning. Faced with the failed test situation, Bob did not explicitly, at least not yet, think about what tasks he might already have scheduled to do this morning. By explicitly identifying the inputs, decision makers have the opportunity to correct faulty information that they may otherwise use to make their decisions. The model encourages decision makers to explore the various kinds of information that they have about a problem and to carefully determine feasible solutions. The model also guides decision makers to consider risks that may be associated with alternative solutions or decisions. Faulty assumptions or incorrect knowledge about the problem environment introduces the risk that a chosen solution will not or cannot succeed in solving the problem.

CASE STUDY

California Bridge Problem

The California Department of Transportation, Caltrans, was very proud of the state's freeway infrastructure, which included thousands of bridges. Before the 1991 earthquake, the Caltrans engineers thought that they had the perfect model for freeway bridges in California. They used concrete, oblong, cylinder columns that they thought would withstand an earthquake of magnitude 8.49 on the Richter scale. The 1991 earthquake, of magnitude 6.2, changed the engineers' understanding of earthquake movement. Instead of moving up and down as engineers previously understood, this quake generated a side-to-side motion, which caused the bridge support columns to shear. The two-minute earthquake in 1991 collapsed many bridges in California, including part of the Oakland Bay Bridge.

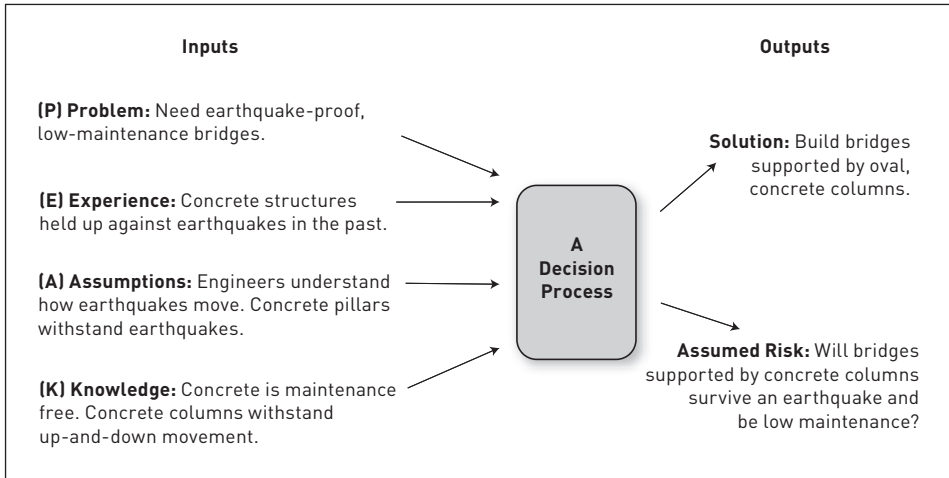


Figure 1-4: *California bridge problem*

Engineers studied the new models of earthquake movement and decided that the solution was to combine two materials to create a new “unbreakable” column. This column has the same concrete oblong structure in the middle as before but now has a steel sleeve on the outside. The steel sleeve holds the concrete in place against the shear, while the concrete columns still defend against an up-and-down motion. The problem is that the columns need regular maintenance because the steel sleeves rust. The original concrete columns were basically “maintenance free” (El-Azazy).

This scenario shows that some of the inputs can be based on faulty knowledge. Caltrans wanted a solution to bridge survival in earthquakes that was maintenance free. They based their solution on knowledge and experience that they thought was correct. The new earthquake model forced a new decision with the constraint of required bridge maintenance. The idea that knowledge remains fixed and is correct can create an environment in which the risk associated with decisions based upon this knowledge is unacceptable. As the Caltrans case illustrates, the inputs and outputs to making a decision can and sometimes should change. Figure 1-4 shows the inputs and outputs for the initial decision to build concrete supports.

Inputs

(P) Problem: Develop a mechanism for bridge support that is earthquake-proof and low maintenance.

(E) Experience: Concrete structures withstood earthquakes in the past.

(A) Assumptions:

- Engineers have a good understanding of how earthquakes move.
- Large, concrete pillars will withstand earthquakes.

(K) Knowledge:

- In California, concrete is basically maintenance free (fact).
- Concrete structures will withstand forces caused by the up-and-down movement of earthquakes (fact).

Outputs

Solution: Build round, concrete pillars to support bridges.

Assumed Risk: Will bridges supported by concrete pillars survive earthquakes that exhibit other types of motion? (Engineers may not fully understand earthquake movement.)

After the 1991 earthquake, engineers learned that earthquakes exhibit both up-and-down as well as back-and-forth movement. This knowledge led them to formulate a new solution to the problem of building bridges that can withstand earthquakes. The second solution, which involved using steel sleeves and concrete, came with the cost of maintaining the steel supports. The input to the second bridge decision included the new knowledge about earthquake movement as well as the experience of many California bridges collapsing during the 1991 earthquake.

As shown in its application to the California bridge problem, the decision model reflects a “semiclosed” system. There is a feedback loop from the outcome of the first solution into the knowledge and experience inputs for the revised solution. Some inputs are open ended in that they may be influenced but not determined by the outputs. For instance, the outputs from a decision may lead to but not necessarily constrain future problems. Likewise, the output from a prior decision can affect but not control a decision maker’s assumptions. Decision makers control their own assumptions. The output from a decision may impact the environment (knowledge) of a future problem, but it will not be the sole influence because many changing factors affect the environment for a problem.

CASE STUDY

Unfamiliar Legacy Code Problem

This short story helps illustrate how a decision maker’s awareness of the risk assumed with alternative solutions may influence the decision that is actually made. The example also clarifies the role of the decision model in helping decision makers brainstorm about alternative solutions to a problem.

Iola is inspecting some “ancient” legacy code and discovers some unfamiliar routines. She asks her colleagues whether they know what the routines do, but most people reply that they don’t know. The entire system is too large to rebuild and test as a whole, so she temporarily forgets about the code. But the mysterious code continues to bother her. What if the code performs no useful function and is just wasting space? Maybe she should comment it out or refactor it to be more efficient. She knows that the company is looking for ways to improve legacy code, but removing the code might cause another part of the system to fail or to produce bad results.

Iola decides that the risk of system failure is minimal. Therefore, she decides to comment out the unfamiliar code and to reload the component containing this code into the test environment. She has observed that changes to the other code in the component seem to affect the regression test results, so she assumes that the unfamiliar code will do the same. After three hours of regression testing, Iola detects no failures. She is not sure how to proceed because she has not seen the system respond to the removal of the unfamiliar code. In other words, there has been no smoke from the smoke test. Since the software developers have never built a whole-system regression test, Iola is concerned that she may never be able to know conclusively whether removing the code has a negative impact on the system. Iola decides that the risk of removing the code is too high because she can’t determine whether the removal actually has an impact on the system. She uncomments the code and puts it back into the test system. Figure 1-5 shows the inputs and outputs for Iola’s initial but not final decision.

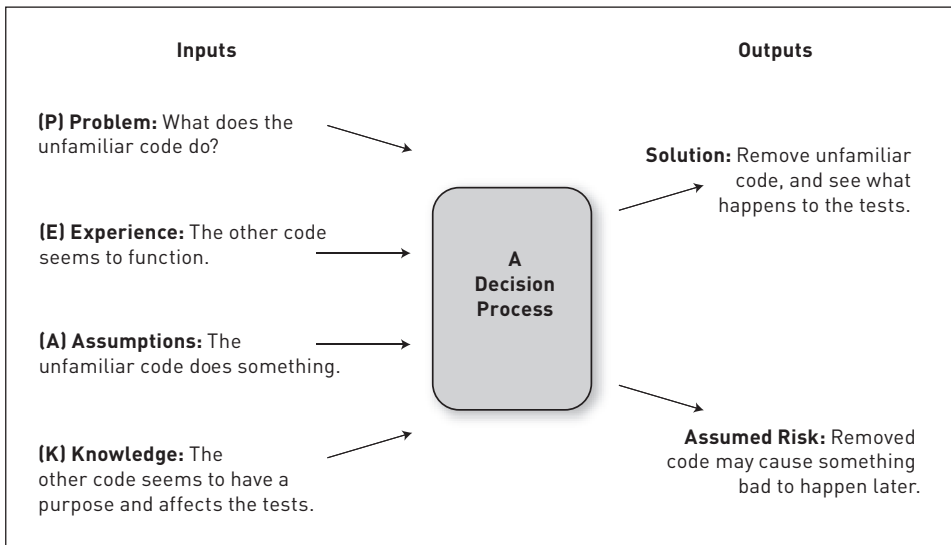


Figure 1-5: Unfamiliar legacy code problem

Inputs

(P) Problem: What does the legacy code do, and how should it be handled?

(E) Experience: Iola understands what most of the other code does and is able to test it.

(A) Assumptions: The unfamiliar legacy code should be there for a reason.

(K) Knowledge:

- The other code in the legacy component seems to have a purpose (fact).
- Other code changes seem to result in test result changes (fact).

Outputs

Solution: Comment out the unfamiliar legacy code, and see what happens.

Assumed Risk: Removing the unfamiliar legacy code may cause something bad to happen.

Through careful examination of inputs to the model, the problem solver may think of alternative solutions to the problem. Since Iola knows that other code in the component seems to affect the regression tests, maybe she could comment out the unfamiliar code and replace it with a message to indicate that the unfamiliar code would now be executing. She might also add some code to log the environment when the message code executes so that she can trace back to the caller of the routine containing the unfamiliar code.

The decision model is a convenient way to encapsulate factors that influence the result of the decision-making process. A model is more comprehensible than an abstract idea because it is a representation of reality. But since a model is not reality itself, it allows decision makers to examine and evaluate aspects of reality without altering them. Lastly, the model provides a frame of reference for studying information that people use to make decisions.

CASE STUDY

Data-Processing Problem

This last example will show how software project managers and software engineers can use the model to help manage their decisions. This story highlights the challenge of determining the real problem to be solved. If they are not currently overwhelmed, people who enjoy solving problems are curious when they hear that there is a problem. They expect the person with the problem to state very carefully all the facts concerning the problem. In the *Star Trek Voyager* television show, the character Doctor asks, “Please state the nature of

the medical emergency” (CBS Studios). The doctor would not be able to help if the character with the problem responded by giving a life history or a synopsis of the next episode of *Days of Our Lives*.

Similarly, a software engineer may receive unhelpful information about a problem but still be expected to provide a solution to the person with the problem. The software engineer may not have a clear set of facts from which to start. Organizing the information one needs to solve a problem to make a proper decision can be difficult. The decision model will help problem solvers determine whether they have the information needed to determine appropriate solutions.

In particular, the model helps decision makers begin the process of “divide and conquer” in a systematic way. They start by perusing all the information (data) that they have received to find the four decision-making inputs. First, they ask themselves whether they have sufficient information to clearly identify the problem or issue that needs a solution. If they do, then they work on the other inputs needed for the decision. If they cannot clearly identify the real problem, then they will not be able to determine an appropriate solution.

In this example, an engineer came to the head of an engineering data-processing organization with a request. He wanted the latest and best-performing computer. The manager asked him why he needed the machine. At the time, the cost of this machine was at least seven times that of the standard machine being used within the organization. The engineer said that his processing system was too slow and was therefore causing him to fall behind in his analysis of data. He thought that the advanced machine would be five times faster than his current processor.

The manager knew the machine that the engineer wanted and agreed about its excellent performance, but she asked whether the engineer knew for certain that the processor was the cause of the slowness. The manager then asked the engineer whether the network could be the problem and explained that a faster data-processing machine would not help if the network was the bottleneck. The manager said that she could give the engineer three computers of the standard type today and that it would probably take 30 days for the new, nonstandard system to arrive.

The manager worked with the engineer and the networking people to determine whether the network was the bottleneck. They found that the network was responsible for the slowness in processing data needed for reports. With this new knowledge, the engineer explored ways to use the network more efficiently and discovered that he could improve the throughput of processed data by using multiple, standard machines working on parts of the data in parallel.

The story illustrates that software professionals need to clearly understand their problems as well as the other inputs before applying any decision process. If the inputs to the decision process are invalid, then the output is most likely to be an invalid solution. Software professionals intuitively know this, but because of time and other constraints, they do not necessarily keep this in mind when they are solving problems. The decision model helps software professionals separate issues and ensure that they do not end up with “garbage in, garbage out.”

Figure 1-6 shows the inputs and outputs for the engineer’s initial solution.

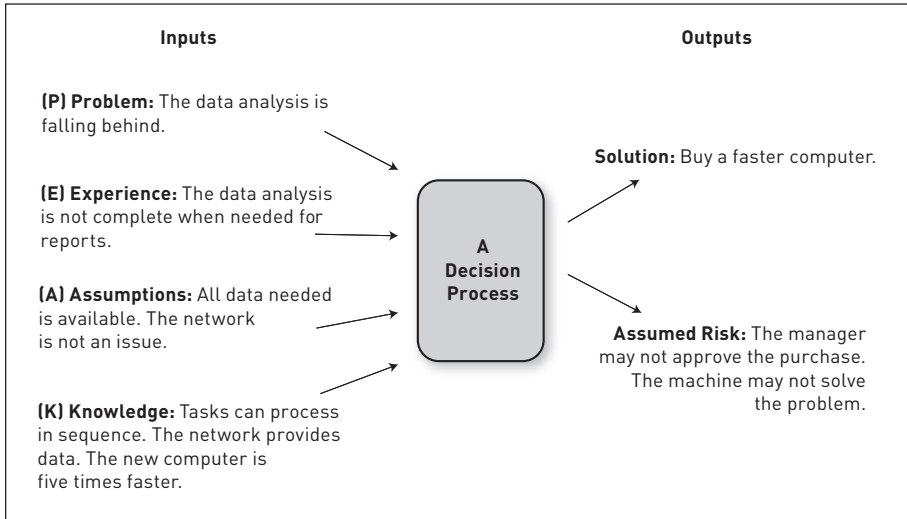


Figure 1-6: Data-processing problem, initial solution

Before Discussion with the Manager

Initial Inputs

(P) Problem: The engineer’s data analysis is falling behind.

(E) Experience: The engineer has not been able to get the data analysis completed when needed for reports.

(A) Assumptions:

- All the data is available to be processed when needed.
- The network is providing the data as needed.

(K) Knowledge:

- The tasks are currently processed in sequence (fact).
- The network transfers data to be processed for reports (fact).
- The engineer sees an advertisement for a computer that is five times faster than the one he currently has (fact).

Initial Outputs

Solution: Buy the new machine that will process data five times faster.

Assumed Risk:

- The software engineer may not receive permission to buy the machine.
- The machine may not solve the engineer’s processing problems.

Figure 1-7 shows the inputs and outputs for the engineer's final solution.

After Discussion with Manager

The engineer made the following changes to his decision inputs after talking with his manager.

Inputs

(P) Problem: The engineer's data analysis is falling behind.

(E) Experience: Multiple machines working in parallel can use network more efficiently.

(A) Assumptions: A number of tasks in the report process can be done in parallel.

(K) Knowledge:

- The network is the bottleneck. (The network was responsible for the slowness in processing data needed for reports.)
- While the network transfers data, it is possible to process report data. (Although many tasks are worked on in sequence, some can be processed in parallel.)

Outputs

Solution: Use a number of in-stock computers to work on parallel parts of the report at the same time.

Assumed Risk: The report processing tasks may not be sufficiently parallel.

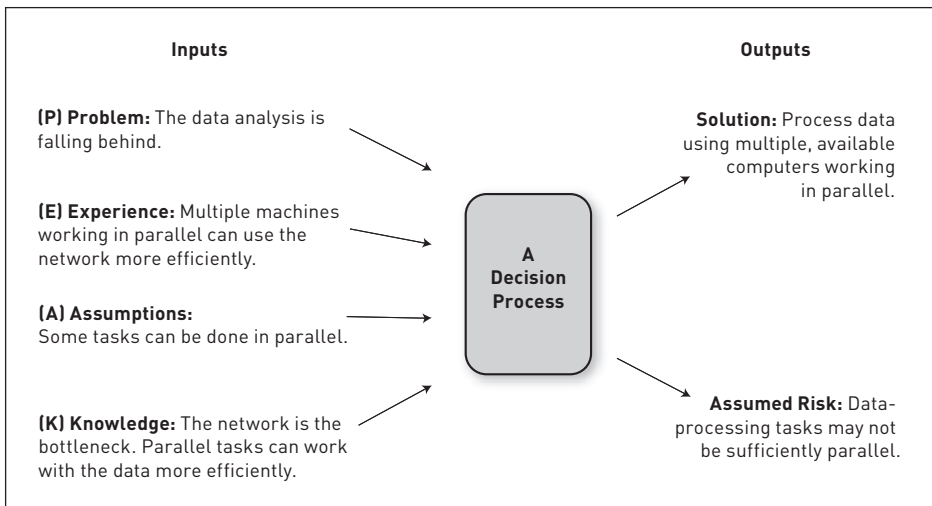


Figure 1-7: *Data-processing problem, final solution*

1.4 Summary

As shown in the example problems, the decision model presented in this chapter does not define your decision process. The model helps you better manage the inputs and outputs to your decision processes and thereby better evaluate your decisions. The case study analyses in the book will show how software professionals can achieve consistently good decisions by verifying that the inputs to their decision processes are valid and correct. You should be aware that there is no guarantee that decisions generated using the model will have successful results. Any one of the following events could occur:

- The decision makers may not apply the model correctly.
- The stakeholders may not implement solutions or decisions correctly.
- The decision makers may not understand reality and therefore may not identify valid or correct inputs.

If the results of the solutions or decisions generated using the decision model do not turn out as planned, the decision makers should review the process they used to make the decision to find any flaws in their understanding of the problem and other inputs. The decision model provides software project stakeholders with a way to analyze the results of their solutions or decisions and to check their understanding of how to manage the decision process. Software professionals can also use the model to reflect on decisions with good outcomes as well as those with poor results. Over time, software project managers and engineers will develop “patterns of thinking” about what constitutes a good decision for specific types of software project management and engineering problems. Knowledge acquired through reflection on past decisions can feed back into future decision making.

The book demonstrates how to apply the decision model to decisions in nine key areas of software engineering. Each chapter focuses on one key area and shows how the model applies to decision making to manage software projects in this area. Though its purpose is to help you make successful decisions, the book cannot model every software engineering decision that you will encounter. Therefore, we have selected the “high-value” decisions that frequently make or break software projects.

Lastly, we must emphasize that a decision process is individualistic. The purpose of this book is to help you better evaluate your own decision processes. The idea is for you to use the concepts presented in this chapter to model the inputs and outputs of your decision processes. Different people might need more or less of the inputs for specific problems, but all decision makers will need some of these inputs to develop solutions. Finally, we want you to understand the concept of risk associated with managing decisions. Managing decisions by managing the risk associated with these decisions is essential to successful software project management.

Index

A

- Actual cost of work preformed (ACWP), 110
- Agile software development process
 - Bank on the Verge: Case Study, 176, 183
 - Managing Global Software
 - Development at FibOptia: Case Study, 332, 337
- Algorithmic (model-based) estimation
 - decision approaches for PEAK, 72
- Aligning process objectives with business goals, 170
- Allocating overhead, 69
- ARCEP (Autorité de Régulation des Communications Électroniques des Postes), 313
- Assumed risk
 - Bank on the Verge: Case Study, 187
 - California bridge problem, 12
 - change requests, 35
 - managing global development, 328
 - managing plans, 97
 - managing process, 173
 - managing stakeholder expectations, 288
 - New Technology—Is It Always the Best?: Case Study, 157
 - On Time, Within Budget, but Wrong: Case Study, 51
 - PEAK decision-making model, 6
 - people interactions, 255
 - schedules, 145
 - SEWeb and Russoft Technologies: Case Study, 221
 - To Replan or Not to Replan?: Case Study, 123
 - when a team runs a race, 90
- Assumptions
 - Bank on the Verge: Case Study, 186
 - California bridge problem, 11
 - change requests, 34
 - estimating cost, 74
 - estimating quality, 76
 - estimating schedule, 76
 - estimating scope, 75
 - estimation decision approaches for PEAK, 73
 - managing global development, 327
 - managing plans, 96
 - managing process, 173
 - managing stakeholder expectations, 286
 - New Technology—Is It Always the Best?: Case Study, 157
 - PEAK decision-making model, 6
 - schedules, 145
 - SEWeb and Russoft Technologies: Case Study, 220
 - software test rerun problem, 9
 - To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 254
 - To Replan or Not to Replan?: Case Study, 122
- ATM (automatic teller machine), Bank on the Verge: Case Study, 177
- ATRAM project, Damn the Process, Full Speed Ahead: Case Study, 191
- Attributes. *See* Quality attributes; Software quality attributes
- Automatic teller machine (ATM), Bank on the Verge: Case Study, 177
- Autorité de Régulation des Communications Électroniques des Postes (ARCEP), 313

B

- Bank on the Verge: Case Study, 175-188
- BCWP (budgeted cost of work performed). *See* Earned value (EV)
- BCWS (budgeted cost of work scheduled). *See* Planned value (PV)
- Bias, estimation bias, 84
- Bottom-up resource allocation, managing estimates, 69
- Brake feedback monitor (BMF), 149
- Bridge problem case study, 10
- Budget
 - contract elements, 31
 - customer expectations, 272
 - decisions to make when establishing requirements, 25
 - decisions to make when managing requirements change and scope, 36
 - decisions to make when managing software project plans, 103
 - estimating, 73
 - guidelines when making decisions about project plans, 107
 - Managing Plans Is in the Details: Case Study, 133
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - software project planning, 99
 - To Replan or Not to Replan?: Case Study, 109-125
 - tracking, controlling and estimating, 102
- Budgeted cost of work performed (BCWP). *See* Earned value (EV)
- Budgeted cost of work scheduled (BCWS). *See* Planned value (PV)
- Budget planning, managing plans, 101
- Business goals
 - aligning process objectives with, 170
 - decisions when establishing requirements, 24
 - On Time, Within Budget, but Wrong: Case Study, 54-65

Business objectives, aligning requirements with, 23

Buy-in from rank-and-file engineers, Bank on the Verge: Case Study, 175-188

C

- Calendar synchronization, 309
- California Bridge Problem: Case Study, 10
- Case studies
 - Bank on the Verge, 175-188
 - California Bridge Problem, 10
 - Damn the Process, Full Speed Ahead, 189-193
 - Data-Processing Problem, 14-17
 - Estimation as a Tool, 78-83
 - Falcon Edutainment and the RiskSim Project, 223-231
 - Friend or Foe at Hanover-Tech, 258-264
 - Globally Distributed Team: FibreNet Project, 311-330
 - Managing Global Software Development at FibOptia, 330-341
 - Managing Plans Is in the Details, 125-137
 - New Account Project at HBC, 39-54
 - New Technology—Is It Always the Best?, 147-159
 - On Time, Within Budget, but Wrong, 54-65
 - SEWeb and Russoft Technologies, 201-222
 - Software Test Rerun Problem, 7-10
 - TCP enhancements at Gigaplex Systems, 275-292
 - To Be or Not to Be: A Sense of Urgency at TestBridge, 243-257
 - To Replan or Not to Replan?, 109-125
 - Tough Sell at Henkel Labs, 292-304
 - Unfamiliar Legacy Code Problem, 12
 - When a Team Runs a Race, 84-91
 - Why is This Product Wrong?, 159-165

- Change management
 - estimates, 70
 - New Technology—Is It Always the Best?: Case Study, 147-159
- Changes in requirements
 - about, 33-37
 - defined, 22
- Changes to process midstream, 169
- Changes to project plans, managing in midstream, 102
- Client expectations, TCP Enhancements at Gigaplex Systems: Case Study, 275-292
- Collaboration
 - Globally Distributed Team: FibreNet Project: Case Study, 311-330
 - global software development, 305
 - Managing Global Software Development at FibOptia: Case Study, 330-341
- Communications
 - with customer stakeholder, 270
 - Friend or Foe at Hanover-Tech: Case Study, 258-264
 - Globally Distributed Team: FibreNet Project: Case Study, 311-330
 - importance of, 233
 - Managing Global Software Development at FibOptia: Case Study, 330-341
 - managing people interactions, 233-265
 - stakeholder interactions, 234
 - To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
 - Tough Sell at Henkel Labs: Case Study, 292-304
- Communications about risk, 195, 200, 223-231
- Communication style, influencing people interactions, 237
- Compensation, To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
- Conceptual design of a solution, managing estimates, 69
- Cone of uncertainty, when a team runs a race, 70
- Configuration management, activities in managing product, 142
- Conflict management, interactions between project stakeholders, 236
- Connectionless communication, defined, 314
- Conquer (bottom-up resource allocation), managing estimates, 69
- Consistency, decisions to make when eliciting and stating requirements, 27
- Context situation, influencing people interactions, 237
- Contingencies, contract elements, 31
- Contracting requirements
 - about, 30-33
 - defined, 22
- Controlling
 - budgets, 102
 - quality assurance, 142
 - risk, 199
- Conversion projects, 295, 298, 300
- Cost
 - contract elements, 31
 - customer expectations, 272
 - decisions to make when establishing requirements, 25
 - decisions to make when managing requirements change and scope, 36
 - estimating, 73
 - key concepts, 144
 - managing process, 173
 - risk management, 197
 - stakeholders' point of view, 3
- Cost performance index (CPI)
 - defined, 111
 - To Replan or Not to Replan?: Case Study, 118

- Coverage, decisions to make when
 - eliciting and stating requirements, 27
 - Criteria, for verification and validation, 29, 38
 - Cultural challenges, Globally Distributed Team: FibreNet Project: Case Study, 311-330
 - Cultural differences, Globally Distributed Team: FibreNet Project: Case Study, 323
 - Culture, tradition and mores, influencing people interactions, 237
 - Customer communications
 - managing stakeholder expectations, 270
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - Customer expectations
 - Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - managing, 223-231, 272
 - TCP Enhancements at Gigaplex Systems: Case Study, 275-292
 - Customers. *See also* Stakeholders
 - On Time, Within Budget, but Wrong: Case Study, 54-65
 - stakeholder expectations, 268, 271
 - types of, 271
 - Customer satisfaction, factors in, 274
 - Customer-user-manager entity, 268
 - Customer-user-team-boss entity, 268
 - CyCorp, TCP Enhancements at Gigaplex Systems: Case Study, 275-292
- D**
- Damn the Process, Full Speed Ahead: Case Study, 189-193
 - Data-Processing Problem: Case Study, 14-17
 - Decision-making
 - managing decisions, 1-18
 - managing estimates, 67-92
 - managing global development, 305-342
 - managing people interactions, 233-265
 - managing plans, 93-138
 - managing process, 167-194
 - managing product, 139-165
 - managing requirements, 19-66
 - managing risk, 195-232
 - managing stakeholder expectations, 267-305
 - PEAK decision model, 4-7
 - Defecting customers, 271
 - Defining
 - process, 170
 - products, 142
 - project objectives, 100
 - Deliverables
 - contract elements, 31
 - defined, 23
 - validating, 37
 - Delivery, products, 141
 - Delivery schedules, When a Team Runs a Race: Case Study, 84
 - Department of Defense (DoD), On Time, Within Budget, but Wrong: Case Study, 54
 - Desirement, customer expectations, 272
 - Developers, stakeholder expectations, 268
 - Development, products, 141
 - Development management, activities in
 - managing, 142
 - Directness, personality styles, 240
 - Disclaimers, contract elements, 31
 - Distributed software development, 306
 - Divide (element-wise decomposition),
 - managing estimates, 69
 - DoD (Department of Defense), On Time, Within Budget, but Wrong: Case Study, 54
- E**
- Earned value (EV), defined, 111
 - Efficiency, managing process, 172

- Element-wise decomposition, managing estimates, 69
 - Eliciting requirements, 21, 25
 - Employee compensation, To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
 - Employee motivation, To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
 - Engineering. *See* Software
 - Environment, influencing people interactions, 237
 - Establishing requirements
 - about, 23
 - defined, 21
 - Estimates, 67-92
 - budget, 73, 102
 - case studies about, 77-91
 - change management, 70
 - construction, 70
 - defined, 68
 - global software development, 307
 - quality, 76
 - schedules, 75, 102
 - scope, 74
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - Estimation as a Tool: Case Study, 78-83
 - Estimation bias, When a Team Runs a Race: Case Study, 84
 - EV (earned value), defined, 111
 - Evaluating
 - process performance, 171
 - process results, 169
 - project plans, 106
 - ExecSoft, Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - Execution, managing process, 169
 - Expectations
 - Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - influencing people interactions, 237
 - stakeholders, 257, 267-305, 309
 - To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
- F**
- Face-to-face communications, global software development, 316
 - Factor influence analysis, upcoming communications, 241
 - Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - Faster-paced personalities, 240
 - Fault tolerance, defined, 190
 - Fiber-optic communications, Globally Distributed Team: FibreNet Project: Case Study, 312
 - FibOptia case study, 330-341
 - FibreNet Project case study, 311-330
 - Field service, activities in managing product, 143
 - Formats, decisions to make when eliciting and stating requirements, 28
 - FTTH (fiber to the home), 313, 319
 - Functional requirements
 - decisions to make when eliciting and stating requirements, 27
 - decisions to make when establishing requirements, 24
 - defined, 20
- G**
- Gantt chart, To Replan or Not to Replan?: Case Study, 114
 - Geographical Information System (GIS), defined, 55
 - Geographical Navigational Satellite System (GNSS), defined, 55
 - Gigaplex Systems Case Study, 275-292
 - GIS (Geographical Information System), defined, 55

Global development projects, managing, 305-342

Globally Distributed Team: FibreNet Project: Case Study, 311-330

Global positioning system (GPS), On Time, Within Budget, but Wrong: Case Study, 54

Global software development, defined, 305

GNSS (Geographical Navigational Satellite System), defined, 55

Goals. *See also* Business goals; Business objectives; Project objectives influencing people interactions, 237

Goal setting, To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257

GPS (global positioning system), On Time, Within Budget, but Wrong: Case Study, 54

Greenfield development, defined, 148

Guardedness, personality styles, 240

H

Hamilton Banking Corporation (HBC), new account project case study, 39-54

Henkel Labs Case Study, 292-304

Historical data, estimation decision approaches for PEAK, 72

Hostages (customer type), 271

I

Identifying, risks, 199

Incremental software release, defined, 313

Indirectness, personality styles, 240

Information technology (IT), New Account Project at HBC: Case Study, 40

In-person meetings, 307, 309, 318, 325, 327

Inputs to PEAK decision-making model about, 6
California bridge problem, 11

data-processing problem, 16

software test rerun problem, 9

unfamiliar legacy code problem, 14

Installation, activities in managing, 142

Interaction size, influencing people interactions, 237

Internet communications protocols, 281

Internet protocol (IP), 313

IT (information technology)
New Account Project at HBC: Case Study, 40
To Replan or Not to Replan?: Case Study, 110

Iterative software development, defined, 313

L

Leadership, interactions between project stakeholders, 236

Legacy Code Problem: Case Study, 12

Legal issues, decisions to make when establishing requirements, 25

Life-cycle development, Bank on the Verge: Case Study, 175-188

Loaded cost, defined, 85

Loyal customers, 271

M

Managing
customer expectations, 223-231, 272
decisions, 1-18
estimates, 67-92
expectations, 258-264
global development, 305-342
people interactions, 233-265
perceptions, 271
plans, 93-138
process, 167-194
product, 139-165
requirements, 19-66, 201-222
risk, 195-232

- stakeholder expectations, 267-305
 - stakeholder relationships, 258-264
 - Managing Global Software Development at FibOptia: Case Study, 330-341
 - Managing Plans Is in the Details: Case Study, 125-137
 - Managing requirements, defined, 20
 - MDSO (model-driven software development), verifying and validating requirements, 29
 - Measuring, process, 171
 - Mercenaries (customer type), 271
 - Metrics, defined, 79
 - Milestone planning, managing plans, 100
 - Milestones
 - defined, 98
 - guidelines when making decisions about project plans, 106
 - Mitigating risk, 199
 - Model-based estimation decision approaches for PEAK, 72
 - Model-driven software development (MDSO), verifying and validating requirements, 29
 - Models. *See also* PEAK decision-making model
 - stakeholder expectations model, 2
 - Motivation
 - influencing people interactions, 237
 - To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
- N**
- Negotiating requirements
 - about, 30-33
 - defined, 22
 - Negotiations
 - between project stakeholders, 236
 - To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257
 - Tough Sell at Henkel Labs: Case Study, 292-304
 - New Account Project at HBC: Case Study, 39-54
 - New Technology—Is It Always the Best?: Case Study, 147-159
 - Non-functional requirements
 - decisions to make when eliciting and stating requirements, 27
 - decisions to make when establishing requirements, 24
 - defined, 20
 - On Time, Within Budget, but Wrong: Case Study, 60
- O**
- Objectives. *See also* Business objectives; Project objectives
 - influencing people interactions, 237
 - Object-oriented (OO)
 - Bank on the Verge: Case Study, 182
 - defined, 148
 - Offshore development
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - Tough Sell at Henkel Labs: Case Study, 292-304
 - On Time, Within Budget, but Wrong: Case Study, 54-65
 - OO (object-oriented)
 - Bank on the Verge: Case Study, 182
 - defined, 148
 - Openness, personality styles, 240
 - Open System Interconnection (OSI) Reference Model, 313
 - Outputs to PEAK decision-making model
 - about, 6
 - California bridge problem, 12
 - data-processing problem, 16
 - software test rerun problem, 9
 - unfamiliar legacy code problem, 14
 - Outsourcing, 306
 - Overhead, allocating, 69

P

- Packet-switched network, defined, 314
- PEAK decision-making model
 - about, 4-7
 - California bridge problem, 11
 - change requests, 34
 - data-processing problem, 16
 - Estimation as a Tool: Case Study, 82
 - managing estimates, 72
 - managing global development, 326
 - managing plans, 95
 - managing process, 168, 173
 - people interactions, 253
 - product management, 140
 - risk management, 196
 - schedules, 145
 - software test rerun problem, 8
 - unfamiliar legacy code problem, 13
- People interactions
 - global software development, 309
 - managing, 233-265
- People-oriented personalities, 240
- Perceptions
 - product, 142, 159-165
 - stakeholders, 271
 - When a Team Runs a Race: Case Study, 84
- Personality, influencing people
 - interactions, 237, 239
- Phone Switch Customer Service (PSCS), 163
- Planned value (PV), defined, 112
- Plans. *See also* Software project plans; Software release plans
 - budget planning, 101
 - defined, 93
 - Globally Distributed Team: FibreNet Project: Case Study, 311-330
 - global software development, 308
 - managing, 93-138
 - PEAK decision-making model, 95
 - schedule planning, 101
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - staffing and other resource
 - planning, 101
 - task and milestone planning, 100
 - types of, 94
- The Platinum Rule: Discover the Four Basic Business Personalities and How They Can Lead You to Success*, 240
- Policy. *See* Project policies
- Price, customer expectations, 273
- "Price is no object," 143
- Priorities
 - estimating scope, 75
 - for stakeholders when negotiating or contracting requirements, 32
- Process, 167-194. *See also* Decision-making; PEAK decision-making model
 - activities of process management, 169
 - as a necessary evil, 170
 - case studies about, 174-193
 - contract elements, 31
 - Damn the Process, Full Speed Ahead: Case Study, 189-193
 - decisions to make when eliciting and stating requirements, 28
 - decisions to make when establishing requirements, 25
 - decisions to make when managing requirements change and scope, 36
 - decisions to make when negotiating or contracting requirements, 33
 - decisions to make when validating software project deliverables, 38
 - decisions to make when verifying and validating requirements, 30
 - defined, 167
 - global software development, 308, 310
 - making decision about, 172
 - Managing Global Software Development at FibOptia: Case Study, 330-341
 - managing stakeholder expectations, 270

- PEAK decision-making model, 168
- product management, 143
- software project planning, 99
- stakeholder expectations model, 2
- versus product issues, 160
- Why Is This Product Wrong?: Case Study, 159-165
- Process improvement, defined, 169
- Process management
 - Bank on the Verge: Case Study, 175-188
 - Damn the Process, Full Speed Ahead: Case Study, 189-193
- Product
 - global software development, 308
 - Managing Global Software Development at FibOptia: Case Study, 330-341
 - managing stakeholder expectations, 270
- Product definition, activities in managing, 142
- Product delivery, 141
 - activities in managing, 142
 - process involvement with, 189
- Product development, PEAK inputs, 140
- Productivity numbers, defined, 79
- Product-line development
 - activities in managing, 142
 - New Technology—Is It Always the Best?: Case Study, 147-159
- Product management
 - defined, 140
 - New Technology—Is It Always the Best?: Case Study, 147-159
 - PEAK decision-making model, 140
 - process, 143
 - Why Is This Product Wrong?: Case Study, 159-165
- Products. *See also* Software products managing, 139-165
- Product support, PEAK inputs, 141
- Product versus process issues, 160
- Project cost. *See* Cost
- Project decision-making
 - managing decisions, 1-18
 - managing estimates, 67-92
 - managing global development, 305-342
 - managing people interactions, 233-265
 - managing plans, 93-138
 - managing process, 167-194
 - managing product, 139-165
 - managing requirements, 19-66
 - managing risk, 195-232
 - managing stakeholder expectations, 267-305
- Project estimation. *See* Estimates
- Project management. *See also* Managing; Project decision-making
 - global software development, 310
 - Tough Sell at Henkel Labs: Case Study, 292-304
- Project management area, project plan information, 98
- Project management data and metrics, Managing Plans Is in the Details: Case Study, 125-137
- Project objectives
 - contract elements, 31
 - managing plans, 100
- Project organization, decisions to make when managing software project plans, 103
- Project planning, SEWeb and Russoft Technologies: Case Study, 201-222
- Project plans. *See also* Software project plans
 - decisions to make when managing software project plans, 109-125
 - guidelines when making decisions about project plans, 105
 - Managing Plans Is in the Details: Case Study, 125-137
 - To Replan or Not to Replan?: Case Study, 109-125
- Project policies, business planning, 100

- Project quality. *See also* Quality; Software quality
 decisions to make when managing software project plans, 104
- Project replanning
 decisions to make when managing software project plans, 105
 To Replan or Not to Replan?: Case Study, 109-125
- Project risks. *See also* Risk
 decisions to make when managing software project plans, 104
 PEAK decision model, 196
 To Replan or Not to Replan?: Case Study, 109-125
- Projects. *See also* Software projects
 defined, 93
 tracking, 105
- Project schedules. *See* Delivery schedules; Schedules
- Project scope. *See also* Scope
 managing plans, 100
- Project staffing. *See* Staffing
- Project status, tracking, 134
- Project tracking. *See* Tracking
- Project vital signs, 106
- PSCS (Phone Switch Customer Service), 163
- Pulse points, projects, 106
- PV (planned value), defined, 112
- Q**
- QoS (quality of service), Globally Distributed Team: FibreNet Project: Case Study, 312-337
- Quality. *See also* Software quality; Software quality attributes
 Bank on the Verge: Case Study, 175-188
 cone of uncertainty, 71
 customer expectations, 273
 Damn the Process, Full Speed Ahead: Case Study, 189-193
 decisions to make when managing requirements change and scope, 36
 estimating, 76
 issues in product management, 143
 New Technology—Is It Always the Best?: Case Study, 147-159
 process, 172
 risk management, 197
 software development process as a quality filter, 168
- Quality assurance, activities in managing, 142
- Quality attributes. *See also* Software quality attributes
 defined, 139
 Why Is This Product Wrong?: Case Study, 159-165
- Quality of service (QoS), Globally Distributed Team: FibreNet Project: Case Study, 312-337
- R**
- Radio frequency identification (RFID), On Time, Within Budget, but Wrong: Case Study, 54
- Rational Unified Process (RUP), defined, 39, 332, 337
- Real-time software systems, response expectations, 26
- Redundancy, defined, 190
- References, 343-354
- Relationship management, interactions between project stakeholders, 236
- Remote software development teams, 306
- Renaissance Project, To Replan or Not to Replan?: Case Study, 114
- Replanning projects
 about, 94
 decisions to make when managing software project plans, 105
 To Replan or Not to Replan?: Case Study, 109-125

- Requirements, 19-66
 - case studies about, 7-17
 - eliciting and stating, 21, 25
 - establishing, 21, 23
 - Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - global software development, 307
 - managing activities, 21
 - managing change and scope, 22, 33-37
 - negotiating and contracting, 22, 30-33
 - SEWeb and Russoft Technologies: Case Study, 208
 - understanding for managing estimates, 69
 - validating deliverables to requirements, 23, 37
 - verifying and validating, 22, 28
 - Requirements definition, defined, 20
 - Requirements engineering, defined, 20
 - Requirements management
 - defined, 20
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - Resource allocation. *See also* Budget; Schedules; Shared resources; Staffing
 - decisions to make when managing software project plans, 103
 - guidelines when making decisions about project plans, 107
 - Managing Plans Is in the Details: Case Study, 133
 - tradeoffs when managing risk, 200
 - Resource planning, managing plans, 101
 - Reuse, defined, 148
 - RFID (radio frequency identification),
 - On Time, Within Budget, but Wrong: Case Study, 54
 - Risk. *See also* Assumed risk
 - activities in managing, 198
 - communication about, 195, 200
 - decisions to make when negotiating or contracting requirements, 33
 - decisions to make when verifying and validating requirements, 30
 - Falcon Edutainment and the RiskSim Project: Case Study, 223-231
 - global software development, 309
 - identifying, 199
 - managing, 195-232
 - mitigating, tracking and controlling, 199
 - PEAK decision-making model, 6, 196
 - SEWeb and Russoft Technologies: Case Study, 201-222
 - software project planning, 99
 - RiskSim Project and Falcon Edutainment case study, 223-231
 - Risk statement formulation, 199
 - Roles, stakeholders when negotiating and contracting requirements, 32
 - RUP (Rational Unified Process), defined, 332
 - Russoft Technologies and SEWeb case study, 201-222
- S**
- SA (Selective Availability), defined, 56
 - SAP software, To Replan or Not to Replan?: Case Study, 109
 - Schedule performance index (SPI)
 - defined, 112
 - To Replan or Not to Replan?: Case Study, 118
 - Schedule planning, managing plans, 101
 - Schedules. *See also* Delivery schedules
 - contract elements, 31
 - customer expectations, 273
 - decisions to make when establishing requirements, 25
 - decisions to make when managing requirements change and scope, 36
 - decisions to make when managing software project plans, 103
 - estimating, 75

- Schedules, *continued*
 guidelines when making decisions
 about project plans, 107
 Managing Plans Is in the Details: Case Study, 133
 process, 172
 product management, 144
 To Replan or Not to Replan?: Case Study, 109-125
 tracking, controlling and estimating, 102
- Scope. *See also* Project scope
 cone of uncertainty, 71
 customer expectations, 272
 estimating, 74
 of requirements, 22, 33-37
 risk management, 196
- Scope creep, change requests, 35
- Scrum, defined, 277
- Selecting, process, 170
- Selective Availability (SA), defined, 56
- Self management, interactions between project stakeholders, 236
- Separation, of stakeholders, 30
- Service. *See also* Field service; Software service
 about, 143
- SEWeb and Russoft Technologies: Case Study, 201-222
- Shared resources, Managing Plans Is in the Details: Case Study, 125
- Sharing information, Globally Distributed Team: FibreNet Project: Case Study, 311-330
- Short time frame, SEWeb and Russoft Technologies: Case Study, 201-222
- Simulation/training sector, Falcon Edutainment and the RiskSim Project: Case Study, 223-231
- Slack time, defined, 112
- Slower-paced personalities, 240
- Small budget, SEWeb and Russoft Technologies: Case Study, 201-222
- Software. *See also* Agile software development process
 decision-making model, 5
 defined, 97
 global software development, 305
 iterative software development, 313
 V model of software development, 314
- Software artifacts, defined, 97
- Software development process, as a quality filter, 168
- Software estimates, defined, 68
- Software evolution, quality attributes, 26
- Software products. *See also* Products defined, 97
- Software professionals, knowledge of decision-making process, 2
- Software project deliverables. *See also* Deliverables defined, 97
 validating, 37
- Software project manager, stakeholder expectations, 268
- Software project plans. *See also* Plans about, 98
- Software projects. *See also* Project decision-making defined, 98
 risk, 196
- Software quality
 DoD definition, 56
 software attributes, 20
- Software quality attributes. *See also* Quality attributes about, 20
 DoD definition, 56
 On Time, Within Budget, but Wrong: Case Study, 60
 software evolution, 26
 software quality, 20

- Software release plans
 - about, 26
 - On Time, Within Budget, but Wrong: Case Study, 56
 - Software service, defined, 98
 - Software Test Rerun Problem; Case Study, 7-10
 - SPI (schedule performance index)
 - defined, 112
 - To Replan or Not to Replan?: Case Study, 118
 - Staffing
 - contract elements, 31
 - decisions to make when establishing requirements, 25
 - decisions to make when managing software project plans, 103
 - guidelines when making decisions about project plans, 107
 - Managing Plans Is in the Details: Case Study, 133
 - planning, 101
 - To Replan or Not to Replan?: Case Study, 109-125
 - Stakeholders
 - communications about risk, 200, 223-231
 - communication with, 233
 - decisions to make when managing requirements change and scope, 36
 - decisions to make when managing software project plans, 103
 - decisions to make when negotiating or contracting requirements, 32
 - decisions to make when validating software project deliverables, 38
 - decisions to make when verifying and validating requirements, 29
 - defined, 267
 - eliciting and stating requirements, 25
 - establishing requirements, 23
 - expectations, 257, 267-305, 309
 - Friend or Foe at Hanover-Tech: Case Study, 258-264
 - managing expectations, 267-305
 - model of expectations, 2
 - New Account Project at HBC: Case Study, 39-54
 - perceptions, 271
 - product management decisions, 148
 - separation of, 30
 - software project plans, 99
 - Storing information, Globally Distributed Team: FibreNet Project: Case Study, 311-330
 - Subplans, creating and working with, 134
 - Subprojects, guidelines when making decisions about project plans, 107
 - Support, products, 141
- T**
- Task-oriented personalities, 240
 - Task planning, managing plans, 100
 - Tasks
 - defined, 98
 - guidelines when making decisions about project plans, 106
 - TCP Enhancements at Gigaplex Systems: Case Study, 275-292
 - Teaching process, 171
 - Team morale, TCP Enhancements at Gigaplex Systems: Case Study, 275-292
 - Team organization, TCP Enhancements at Gigaplex Systems: Case Study, 275-292
 - Teams, remote or virtual software development, 306
 - Testability, decisions to make when eliciting and stating requirements, 27
 - TestBridge case study, 243-257
 - Three-Tech consulting company, 161
 - Threshold of success, risk management, 198

Time

- contract elements, 31
- decisions to make when establishing requirements, 25
- decisions to make when managing requirements change and scope, 36
- estimating, 75
- risk management, 197
- SEWeb and Russoft Technologies: Case Study, 201-222
- slack time, 112

Time separation, Globally Distributed

- Team: FibreNet Project: Case Study, 323

To Be or Not to Be: A Sense of Urgency at TestBridge: Case Study, 243-257

To Replan or Not to Replan?: Case Study, 109-125

Tough Sell at Henkel Labs: Case Study, 292-304

Tracking

- budgets and schedules, 102
- project plans, 105
- projects, 109-125
- project status, 134
- risk, 199
- SEWeb and Russoft Technologies: Case Study, 201-222

Tradition, culture and mores, influencing people interactions, 237

Training, activities in managing product, 143

Training/simulation sector, Falcon

- Edutainment and the RiskSim Project: Case Study, 223-231

Transmission Control Protocol (TCP), 277, 281

Trust

- managing global development, 309, 321, 322, 325, 336, 339
- TCP Enhancements at Gigaplex Systems: Case Study, 283, 291

U

UDP (User Datagram Protocol), 277

Unfamiliar Legacy Code Problem: Case Study, 12

Unified Modeling Language (UML), 39, 332

Unknowns in requirements, identifying and resolving, 29

Use cases, New Account Project at HBC: Case Study, 41

User Datagram Protocol (UDP), 277

Users, stakeholder expectations, 268

V

Validating

- deliverables to requirements, 23, 37
- process, 171
- requirements, 22, 28

Verifying

- process, 171
- requirements, 22, 28

Virtual software development teams, 306

Vital signs, projects, 106

V model of software development, defined, 314, 332

VoIP (voice over packet-switched IP), 314

W

Wabash Project for Transport, 149

Warranties, contract elements, 31

Washington Transit Authority (WTA) project, 79

Web-based system development, SEWeb and Russoft Technologies: Case Study, 204, 207

When a Team Runs a Race: Case Study, 84-91

Why Is This Product Wrong?: Case Study, 159-165

Work planning, software project plans, 99

WTA (Washington Transit Authority) project, 79