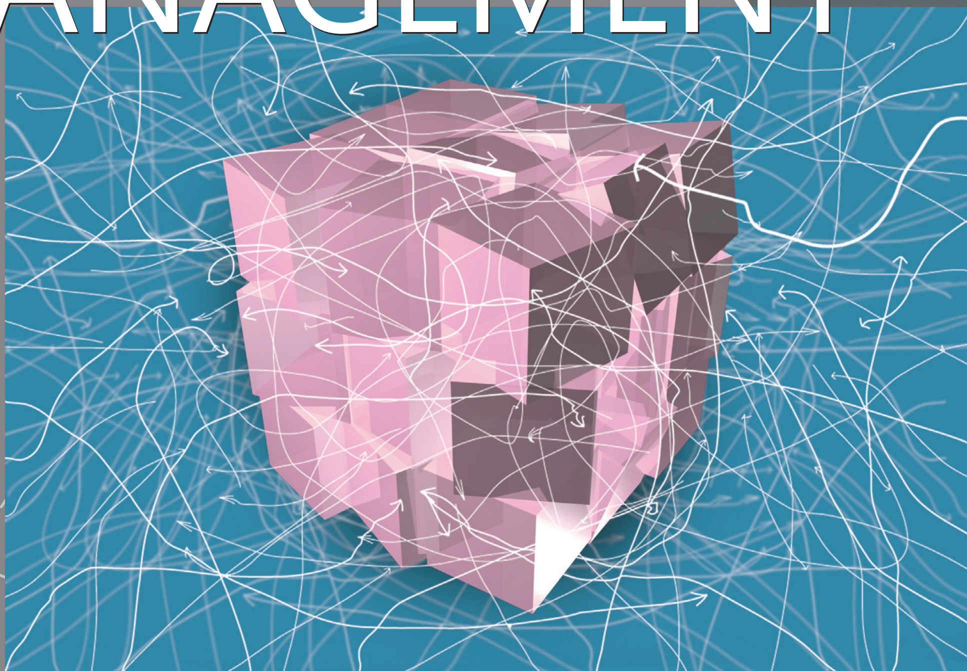
Microsoft

BEST PRACTICES

SOFTWARE CHANGE MANAGEMENT



Case Studies and Practical Advice

Donald J. Reifer

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

Library of Congress Control Number: 2011942372
ISBN: 978-0-7356-6475-3

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at *http://www.microsoft.com/learning/booksurvey*.

Acquisitions Editor: Devon Musgrave
Developmental Editor: Devon Musgrave
Project Editor: Valerie Woolley
Editorial Production: Curtis Philips, Publishing.com
Copyeditor: Roger LeBlanc
Indexer: Lucie Haskins
Cover: Twist Creative • Seattle

Dedicated to those serving as agents of change,
I applaud you.

Contents at a Glance

Contents

Foreword

When you're making decisions about addressing upcoming opportunities and challenges, it is a good practice to do an Analysis of Alternatives. And it is also a good practice to include "Don't change what you're currently doing" as one of the alternatives. In many cases, this can be the best alternative, as reflected in such maxims as "If it ain't broke, don't fix it," "Plan the flight and fly the plan," and "Hold steady on the course."

But there are more and more situations in which "Don't change" is a very risky alternative. A good set of examples is the analysis in Eberhardt Rechtin's book, *Systems Architecting of Organizations: Why Eagles Can't Swim*." Rechtin explains why on six successive satellite system replacement competitions, the more experienced incumbents just did some modifications of their previous winning designs and lost to more innovative competitors. In order to remain competitive in a world of increasing change velocity, you'll want to consider alternatives to "Don't change," and to provide evidence in your Analysis of Alternatives to opponents of change when "Don't change" is not a good idea.

Is the velocity of change all that rapid? In 2006, I published a paper[*] that tried to anticipate future trends and to prepare organizations for accommodating them. By 2011, I found that I had not covered several trends that turned out to be further game-changers for organizations. Most of them were software-intensive: service-oriented cloud computing; social networking technologies; mega-sensor-intensive smart systems; multicore chips requiring software parallelization; and search and mining of ultra-large data aggregations. And the pace of change continues to accelerate, not just in technology, but also in competition and the marketplace. Not only is it important to monitor these changes, but it's at least as important to master the art of making successful organizational change.

This is what makes Don Reifer's book particularly timely and helpful. Getting an organization to change requires getting many stakeholders and suborganizations—who often won't want to change or would like to manipulate the change to increase their power base—to come together, to agree on a mutually satisfactory change strategy, and to contribute their key resources to making the change successful.

Besides Don's technical contributions to such areas as cyber security, cost estimation, business case analysis, software project management, and software maintenance, he has been an effective change agent as a consultant to a remarkably wide variety of

[*] B. Boehm, "Some Future Trends and Implications for Systems and Software Engineering Processes," *Systems Engineering,* Wiley Periodicals, Inc., Volume 9, Issue 1, 2006, pp. 1-19

organizations. These include large and small financial, telecommunications, aerospace, software tools, gaming, and Internet startup companies, and public service organizations. In the book, he provides case studies drawn from these various sectors that illustrate how they have dealt with the need for change to address opportunities or problems at a project, department, business area, or enterprise level.

Along the way, the generally successful case studies illustrate pitfalls to avoid, such as trying to change things outside your span of control, neglecting to provide incentives for change to success-critical stakeholders or leaving them out of the planning process, trying to change too many things in one step, and failing to provide a sound business case for a change initiative.

The diversity of the case studies means that not all of them will be relevant to everyone's situation, but that most people will find some of them to be highly relevant. And the diversity is brought together in the final chapter, which includes ten change management secrets of success; a dozen change management lessons learned; ten useful tools in change management; and summaries of change management critical success factors in dealing with senior managers and in dealing with the workers who will implement the changes. As a bottom line, this book can be very valuable in helping you cope with the increasing pace of change that you'll encounter during your career.

Barry Boehm
September 2011

The phrase "software process improvement" has become a catchphrase for the software industry and occurs hundreds of times in monthly journal articles and also in scores of books. But what does the phrase really mean and how do the concepts get applied in the real world? Don Reifer has been studying software in major companies for many years and has assisted many in improving software development methods and practices.

Process improvement is closely linked to change management. Change in corporations is sometimes glacial and often resisted strongly. Don's book includes some interesting factual information, and also procedural information, about introducing both structural changes and organizational changes that do not disrupt ongoing operations.

At the level of individual projects, change control is also a critical factor. In fact, from my observations while working as an expert witness in software litigation, the two main sources of lawsuits are poor quality control and poor change control.

The measured rate at which software projects change is between 1 and 2 percent per calendar month. For large systems with schedules in the 36 to 48 month range, more than 25 percent of the features that are present at delivery were not there when the requirements were first defined. They came in later due to either incomplete requirements gathering or external business changes that were not predictable.

Don's new book is not a theoretical treatise on change management and software process improvement, but rather it's a series of a dozen empirical case studies from both companies and government groups. The book also covers improvements in both development and maintenance operations.

Software change management and process improvement involve more than mere acquisition of a few tools that support specific methods such as Agile or the Team Software Process. Rather, the issues addressed include a full spectrum of organization topics, methodological topics, tools, and the measurement and reporting of improvement results. In fact, the measurement and reporting of results has been the Achilles heel of many process improvement attempts. The organizations may get better, but if they don't measure the improvements and the costs needed to achieve the improvements then fairly soon top executives will cut off the funding.

Yet another area that needs attention during process improvement activities is the sociological areas of relationships between the information technology (IT) group and its clients, and between the various components of the information technology organization itself.

In many IT shops, the IT world and the client world tend to be adversarial rather than collegial. The adversarial relationships are even worse inside IT groups themselves.

There is often friction between the test community and the development groups, between the maintenance teams and the development teams, and between quality assurance and development teams.

Ordinary corporate politics also play a role, and sometimes projects are canceled because managers don't like each other and refuse to cooperate.

The ten case studies and the shorter anecdotes in Don's book also include some subjects not normally covered in the process literature, such as enabling an academic institute to form a better partnership with information technology companies.

The book provides a very valuable source of empirical data taken from real organizations. The book shows in a step-by-step fashion what the original conditions were, and then the changes that were introduced to improve the initial conditions. These are not trivial changes in small organizations; they are major long-term changes in large and complex organizations.

In general, Don's observations are congruent with my own research on change control and software process improvement. The gist of my findings and the gist of Don's findings are similar:

- Project management is frequently a bottleneck and must be included in all improvements.
- Quality needs to improve first; otherwise, being faster generates more bugs.
- Software defect prevention often needs improvement.
- Pre-test inspections and static analysis often need to be added to quality methods.
- The front end of software projects in requirement and design are often weak links.
- Training and education of all personnel, including management, should be continuous.
- Organization structures are important aspects of process improvements.
- Special care is needed in handling project office, test groups, quality assurance, and other specialist organizations.
- Change control, requirements creep, and deferred features also need to be evaluated.

- Measurement and results are important from day 1 and should become permanent fixtures.
- Executive support is needed, and it requires positive return on investment (ROI) results.
- Processes need to encompass total costs of ownership (TCO).
- Processes need to encompass package acquisition as well as internal development.
- Processes need to encompass contracts and outsourcing as well as internal development.
- Once process improvements occur, new personnel and new managers need to be trained in what the best practices are to ensure continuity.
- The goals of change control and process improvements are closer and more harmonious alignment between business operations and software activities.

Overall, Don's book provides a solid and valuable contribution to the literature on change management and software process improvement methods. It is a book with a very broad focus, and it covers a wide range of topics. This is what the industry needs—not a narrow view of a single method that is claimed to be a panacea.

As Fred Brooks pointed out years ago in *The Mythical Man-Month*, there is no "silver bullet." To get better in software, a wide variety of organizational, social, and technical issues must be addressed in a rational sequence. Don's book adds to this concept and offers a variety of interesting case studies from many organizations.

Capers Jones
President, Capers Jones & Associates LLC
August 2011

Introduction

This book presents ten case studies that revolve around how to manage change in industrial, governmental, and academic settings. Each case was selected to communicate lessons learned that the reader can use to address typical issues that occur during the process of change. Context-sensitive knowledge about how others managed change within these settings is communicated by describing what others did when faced with adversity.

Who Should Read This Book

This book was written to equip those making and managing changes in software organizations with the processes, techniques, and tools that they need to be successful. If you are involved in change initiatives, this book is for you because it points out what the typical issues are that you will face and how others in similar situations have dealt with them.

This book is targeted for consumption by a broad range of readers, from executives to those software engineers who want to pursue change initiatives aimed at getting the job of software development and maintenance done quicker, smarter, and better. Professors will also find this text helpful in communicating the fundamentals associated with instituting and managing change in organizations. Entrepreneurs and business people might want to take advantage of concepts included within the case studies that describe how to facilitate making the changes necessary to transition products to market quicker. Researchers might find the text useful in structuring how they package their new research developments for eventual commercialization.

Assumptions

This book expects that you have at least a basic understanding of underlying software engineering and management fundamentals that set the context for the changes described within the case studies. If you need refresher materials in these topics, you might consider reading Steve McConnell's *Code Complete,* Second Edition (Microsoft Press, 2004), Roger Pressman's *Software Engineering: A Practitioner's Approach,* Seventh Edition (McGraw-Hill, 2009), and Donald Reifer's *Software Management,* Seventh Edition (Wiley/IEEE Computer Society, 2006).

Who Should Not Read This Book

While this book might be interesting reading for entry-level software engineers, such readers need to be warned that the book presents only the background information needed to understand the management structure, industrial practices, implementation issues, and underlying technology for each of the case studies covered. Because the knowledge needed to fully understand the issues more deeply can take years to learn for the uninitiated, these readers and others from non-software backgrounds are warned that some of the discussions on how to resolve problems may be beyond their capacity to fully understand.

Organization of This Book

This book is organized around ten case studies. Chapter 1, "Getting Started," presents some background and context materials for these cases, while Chapter 12, "Making an Impact," provides a summary of lessons learned. The other ten chapters focus on learning experiences presented as case studies that range from making needed organizational changes in a large Information Technology (IT) shop to addressing adoption of Agile methods in a smaller, high technology organization. While based on real-world experiences, all of the cases represent fictitious examples developed to highlight different change management messages. Each of these ten cases is trying to communicate that change is hard and no matter what you do to facilitate the transition to something new, people will resist it. In response, each case tries to highlight the change management principles you can use to make the change and get the job done, often over the objections of others who are more comfortable with the status quo.

Online Companion Content

For those using this text in software engineering courses, I have authored an Instructor's Manual. The purpose of the manual is to help the instructor organize discussions for each of the ten case studies presented in a systematic manner. The manual might also assist others reading the book to determine all of the messages that the cases are trying to communicate. It was fun to write and should be fun to read.

The Instructor's Manual can be downloaded from the following page:

http://www.microsoftpressstore.com/title/9780735664753

Acknowledgments

I would like to first thank my team of peer reviewers, including Bob Charette, Bob Epps, Dr. Ken Nidiffer, and Joan Weszka for their contributions. Their reviews looked at content and made sure the cases presented in the majority of the chapters made sense. I want to next thank Valerie Woolley and her Microsoft Press team for the wonderful job they did editing and preparing the final manuscript. They added a great deal of value by making sure the messages that I tried to communicate in the cases came through by polishing my presentation. Lastly, I would like to thank my family and wife, Carole, who persevered as I wrote this volume. She proofed the early versions of the manuscript and helped me organize my thoughts more coherently.

Errata & Book Support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site:

http://www.microsoftpressstore.com/title/ 9780735664753

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@ microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We Want to Hear from You

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in Touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*

CHAPTER 4

Industrial Case: Moving to Commercial Off-the-Shelf and Open-Source Software Usage in Telecommunications

Setting the stage

This next case study occurs in a large telecommunications firm. The firm wants to move from a custom architecture to an open architecture for its Switching Systems division's product offerings. This division has resisted past attempts to make a move to a new platform and architecture because it had millions of dollars invested in specialized software, which its sales and management leadership viewed as a discriminator in the marketplace. Because of new sales opportunities, the firm initiated the development of a new switch that embraces many innovative concepts. By bringing this switch to market, the firm hopes to retain its market share and position in the future. Everyone in the firm's organizational chain, depicted in Figure 4-1, agrees that it is time to make the changeover and do it right.

The key change being proposed is a move to a new architecture and an open system platform. As shown in Figure 4-2, the applications software will run on top of a POSIX platform riding on top of multiple processors, which execute in parallel to provide growth paths in case more lines need to be added by telephone operating companies (the customers). POSIX will be configured to run using existing facilities to provide platform-designated services on an on-demand basis. Such services include, but are not limited to, configuration and initialization, relational database management, dispatching, distribution, querying, scheduling, and security. Services will run to completion to avoid interrupts that could cause execution to stall, stop, or be rescheduled.

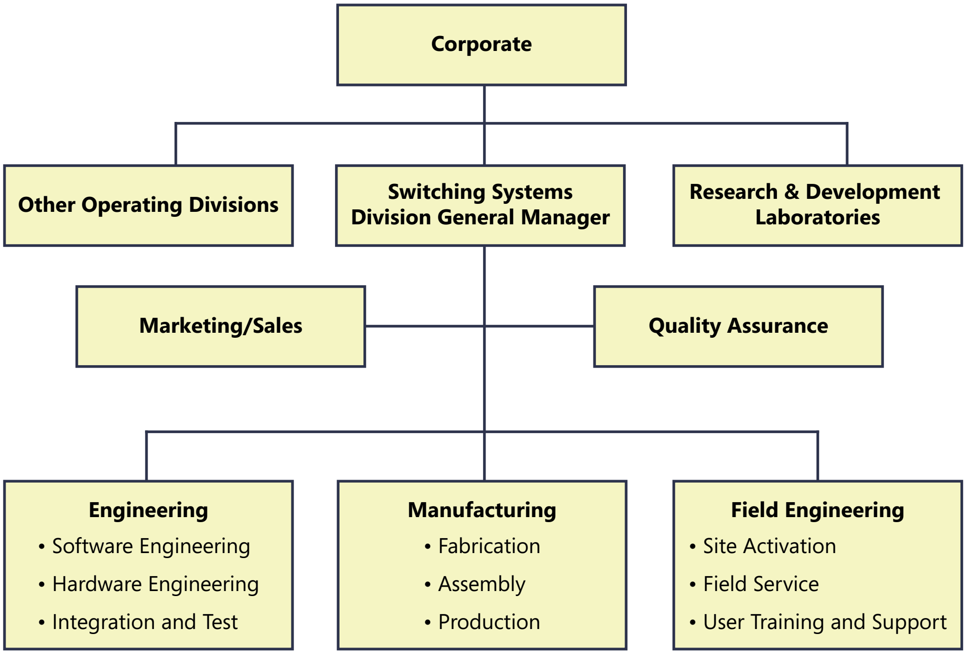


FIGURE 4-1 Telecommunications firm organizational structure.

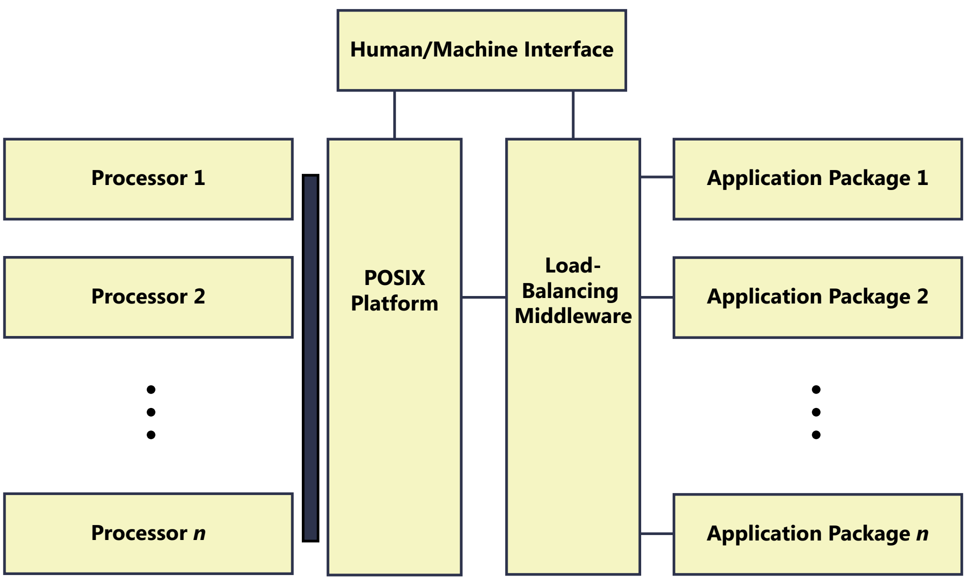


FIGURE 4-2 Top-level switching system architecture.

In addition to using normal switching applications, the new system will provide users with a novel, knowledge-based, self-regulating load-balancing dispatcher and an innovative self-diagnosis and repair system that will act as the marketplace discriminator for sales. The front end of the switch will provide a wide range of network-based and Internet-accessible communications capabilities. It will provide users with easy access to features and query-on-example ability. Context-sensitive help will be provided along with many improved user-interface features to make the system intuitive, easy to understand, and fun to use.

The two key innovative technical enablers that make it feasible to build such a system now are the following: new dispatching algorithms that the Research and Development (R&D) Laboratories invented that facilitate the optimum scheduling of application threads running on parallel processors, and new middleware that allows the system to bind components together using rule-based, load-balancing techniques. Components that are scheduled are fragments of applications packaged by the middleware to execute in parallel on different processors (parallel threads) and share results (self-combinations). Application fragments can include commercial off-the-shelf (COTS) components, open-source or custom routines, modules, or programs, as long as each scheduled and combined entity adheres to the packaging rules, runs under POSIX, and uses the system's data model.

Organization

For this case study, assume that you are the lead software engineer in the Engineering division responsible for developing the new switching system. The initial target of opportunity for sales of the system is a telephone operating company that is your largest customer. This company worked with your people on the architectural specification for the system and helped generate the related functional and performance requirements for it. It wants to buy 100 of these switches, assuming that your organization can deliver them within three years. It is ready to help during the development in any manner possible. The company suggests that it perform the independent verification task, where it provides feedback during the development on the products as they incrementally roll off the drawing board.

Most of the work that the Switching Systems division currently performs is aimed at supporting systems in the field. Major developments like the new system come around once a decade. As such, this represents the means to update the organization's processes, practices, methods, tools, skills, and experience. Management, however, recognizes that by doing too much too quickly your company could fail. In response, they want to attack the development conservatively and use only proven technology. They form a task team to devise a project plan, and you are asked to be a member. You are thrilled and ready to start contributing to the effort.

Project

The project being planned involves the design, development, and validation of a test article that will be used as the model for product development. The development is targeted for three years. Getting manufacturing facilities ready for production will take about year. However, this can be accomplished easily in parallel with the product development because the production facilities are ready for use.

There are no budget details yet. Because the future of the division revolves around the success of this project, you believe management will allocate whatever resources are necessary to pull it off. Even though their funds seem limitless, management wants you to justify every penny.

The planning team is made up of the following six people: you, the team lead, a financial person, the chief engineer, a process person, and a customer representative. Besides several other responsibilities, you have been asked to handle all planning activities associated with COTS and open-source software. Everyone on the team is excited and wants to do a good job.

Your team assessed the current situation and found that both systems requirements and architecture specifications for the new system have been completed by the startup team. These specifications were reviewed as a first order of business and judged to be well done. The team also found that a feasibility study was completed that identified 26 candidate COTS and open-source application software packages for potential use on the project.[1] Several of these candidates have been analyzed on a try-before-you-buy basis, and the results were documented. During development, you know you will still have to select packages, negotiate licenses, and integrate these packages as part of the switching system. You also know that your work with COTS and open-source software will not stop here. There will be annual updates and licensing costs to worry about after the system is operational. Licensing concerns you because there might be run-time license costs associated with some of the packages that have not been accounted for.

Based on this completed work, the team feels much better because their planning efforts would not have to start at square one. In addition, the startup team has completed a high-level budget of $770 million over the three-year development schedule and determined details of the first year's operation in the field, which appear in Table 4-1. As part of your tasking, you are asked to review COTS and open-source forecasts to determine whether or not they are realistic for the job at hand.

TABLE 4-1 Top-level budget for new telecommunications system development and maintenance.

Task	Subtask	Forecasted Budget by Year (in millions)			
		Year 1	Year 2	Year 3	Year 4
Systems engineering	System engineering plan and trade studies	$10	$10	$—	$—
	Integration product team operations	5	10	5	—
Project management	Project management	20	20	20	—
	Measurement and analysis	2	2	2	—
Product support	Configuration management	4	4	4	—
	Quality assurance	4	4	4	—
	Supplier management and licensing	2	2	2	—
	Security and network protection	3	3	3	—
Hardware engineering	Hardware acquisition and readiness	10	10	10	—
	Interface development and test (both hardware and software)	10	10	10	—

Task	Subtask	Forecasted Budget by Year (in millions)			
		Year 1	Year 2	Year 3	Year 4
Software engineering	Requirements analysis	$10	$5	$—	$—
	Software development	25	50	50	—
	COTS and open-source package acquisition and readiness	5	10	10	—
	Software integration and test	10	10	10	—
	Licenses	2	2	2	—
System integration and test	Test planning and readiness	10	—	—	—
	Hardware and software integration and testing	—	25	25	—
	System test and evaluation	—	—	10	—
Manufacturing	Specification	8	3	3	—
	Test article fabrication, assembly, and production	20	25	25	—
	Production article fabrication, assembly, and production	—	25	25	—
Systems test	Test article testing	—	10	10	—
	Acceptance test and evaluation	—	—	10	—
Deployment	Staging and delivery	—	—	25	—
	Dual operations and cutover	—	—	25	—
Operations and maintenance	Planned product improvements (both hardware and software updates and optimizations)	—	—	—	75
	Licenses	—	—	—	5
TOTALS		**$160**	**$240**	**$290**	**$80**

Process

Your next step in the planning process is to determine what work needs to be accomplished to get the product out, determine who will do it, when it will be done, and at what cost. The team lead suggests that the team use a divide-and-conquer strategy to develop the work plan, where each member of the team develops a task list for different parts of the effort. Of course, you are given the COTS and open-source package work as part of your assignment. Your job is to determine whether the line item totals under "Software engineering" titled "COTS and open-source package acquisition and readiness" and "Licenses" are adequate to cover the work required to be completed in these areas.

You first identify the tasks required to employ COTS and open-source packages[2] in the development. Completion of these tasks assumes that the middleware performs as specified and that each package can be cleanly integrated into the system without any rework other than tailoring. The process model used to describe the activities being performed to put COTS and open-source software to work throughout the life cycle is shown in Figure 4-3.

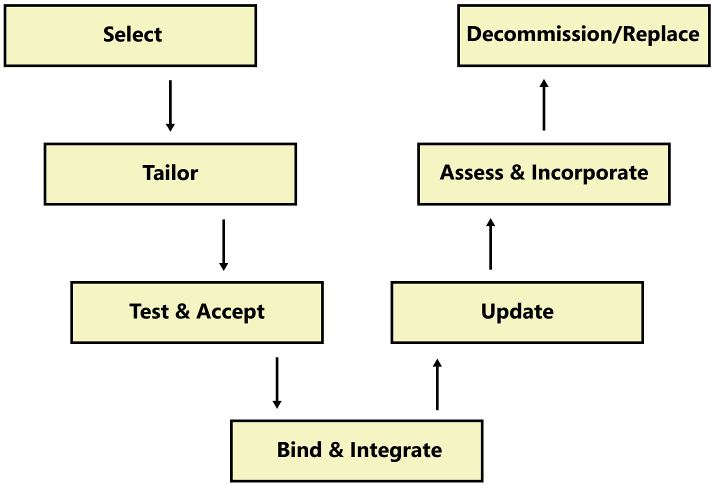


FIGURE 4-3 COTS and open-source package software process model.

For those unfamiliar with terms used in the process model in Figure 4-3, a brief explanation may be in order. After you select a package, you typically configure and tailor it to satisfy your requirements using built-in features. Then you test and accept the package prior to taking delivery. Once you've accepted it, you bind the COTS package via your middleware to your system, integrate it, and make it work with the system at large. Next the vendor provides periodic updates that you evaluate and incorporate into your system, if appropriate. You continue in this mode until you decide it is time to either replace the package with an alternative or decommission it.

The wildcard in the case is the unique data model your people have devised. You must determine whether or not the package can be tailored to accommodate it prior to making your purchase decision. Luckily, this was one of the tasks the startup team performed during the trial licensing period. They assessed the candidate packages to make sure that they worked as advertised and were compatible with the architecture's data-model specifications. You feel relieved when you discover these facts in the notes they provided to you.

Product

You complete your analysis based on the process model shown in Figure 4-3. Based on the selection of the 12 packages listed in Table 4-2, you clearly show that there is more work that needs to be performed than the budget allocated for the packages being considered. The major reason for the cost differential seems to be the run-time licenses for packages for system software (such as the database manager) and tools (such as compilers and debuggers) that operating companies want delivered in case they have to make patches in the field. In addition, as shown in the itemized list in Table 4-2, the budget for COTS and open-source software package licenses is low during software maintenance because many of the development tools and other support licenses were not included in the estimate. The people developing these forecasts overlooked these costs because the costs were not in their frame of reference (software development vs. maintenance).

TABLE 4-2 Itemized costs for COTS and open-source packages.

No.	Package Type	Development License Cost	Maintenance Cost (plus run-time licenses)
1	System software (POSIX toolset)	$250,000	$800,000
2	Database management toolset	$250,000	$2,200,000
3	Requirements management and traceability toolset	$100,000	$100,000
4	Software design toolset	$50,000	$50,000
5	Software language toolset (compiler, and so forth)	$500,000	$1,500,000
6	Software test toolset (coverage, and so forth)	$250,000	$250,000
7	Hardware CAD tools	$500,000	$500,000
8	Software configuration management tools	$250,000	$350,000
9	Test management toolset	$250,000	$250,000
10	Computer performance evaluation toolset	$250,000	$250,000
11	License management toolset	$50,000	$50,000
12	Documentation toolset	$200,000	$200,000
	TOTAL	**$2,900,000**	**$6,500,000**

People

You deliver your findings, and they get incorporated into the project plan. After conducting a peer review, the team lead asks you to prepare some backup materials on COTS and open-source packages because most of the team believes that upper management is clueless when it comes to the issues associated with their use. In response, you develop Table 4-3 to summarize the reasons your firm should buy rather than develop custom software packages, and Table 4-4 to highlight the typical risks associated with the purchase option and related strategies that have been used to successfully mitigate them.[3] These backup charts are received well by the team as the effort to get the plan out concludes.

TABLE 4-3 Developing software in-house vs. licensing it.

Develop Software In-House	License Software
You pay the total development and maintenance cost.	You pay only a fraction of the development and maintenance costs.
Custom software takes years to develop.	Software is available immediately.
The product is mature and relatively bug-free.	It takes considerable time to mature the product.
It's developed to satisfy your customer's requirements.	It's primarily developed to satisfy marketplace requirements.
It's easy to change because you are in charge of the migration path.	It's harder to change because market forces drive the migration path.

Develop Software In-House	License Software
It's hard to determine whether or not the software, when built, will serve the customer's needs.	It's easy to assess what capabilities exist and prototype with existing packages.
Your team must provide user support, training, and documentation for the software.	The package comes with support, training, and user documentation.

TABLE 4-4 Top 10 risks and related mitigation actions taken with licensing software.

No.	Risk	Mitigation Action
1	Hidden license costs.	Understand licenses, and negotiate favorable terms prior to signing the licensing agreement.
2	Package capabilities are not as advertised.	Assess package capabilities via a trial license prior to licensing.
3	Architectural feature mismatches might be present (different data models).	Make sure that you fully assess the package candidates before signing a license agreement.
4	The software architecture puts a premium on performance rather than adaptability.	Modularize the architecture around foreseeable sources of change and the use of COTS.
5	No control over the migration path.	Establish a relationship with the vendor to influence the migration path.
6	Poor customer service.	Pay for on-site support or premium service contract.
7	Most times, all that you get is the executable, not the source code.	If it's really critical, negotiate for a source-code license.
8	The vendor might go out of business.	Establish a market watch to identify replacements, and get the source code put in escrow in case of vendor default.
9	Software upgrades are not in synch with your update cycles.	Architect products using COTS and open-source packages to accommodate such updates.
10	Better alternatives appear on the market, but you are locked into this vendor.	Maintain flexibility in licensing, and keep a market watch.

Options, recommendations, and reactions

You believe that you have everything you need to ask for more money to cover the tasks associated with the use of COTS and open-source software on the development project. You have a rationale for using packaged software, a risk mitigation plan, a package selection list, plus a rationale for your choices, a task list for the work involved, and an associated cost estimate.

You are absolutely shocked when comments are received on the plan. Reviewers from the engineering department sliced your package selection rationale apart. They provided a more technical analysis that justified the use of custom rather than COTS and open-source packages for everything except the system software. They argue in detail that the risks associated with using someone else's software are unacceptable because of performance and security issues. They further suggest that COTS and open-source software cannot be segmented easily to run on parallel processors even with their new scheduling and dispatching software. They point to vendor websites and blogs to provide evidence of security vulnerabilities and breaches that occurred in most of the packages. Based on the input, there is no doubt in anyone's mind that they want to develop the software in-house. When you talk with the team to find out why, you get a big surprise. Several teammates tell you the reason is that the engineering department has a declining workload. They believe the department is using the new development project to justify the jobs for hundreds of people who will be laid off if packaged software is used for the new system.

You are caught in a quandary. You believe that going with COTS and open-source software is the right decision, but you do not want to be responsible for layoffs. The team lead understands your dilemma and suggests that the decision be elevated to senior management. According to the grapevine, engineering department leadership is already working the issue and is using their influence to squash the use of COTS and open-source software in the new system.

You decide the best way to compare the custom options against the COTS options is to look at the time and effort required to complete them. Then, if management wants to ignore the facts, that is their decision. You develop Table 4-5 to make the results easy to comprehend. For simplicity, you include only the software-related development costs across the three-year project schedule. To develop a credible development estimate, you use the parametric COCOMO II cost model[4] using the standard estimating practices endorsed by the firm. Size estimates are those that engineering developed while performing a study to determine how many processors would be needed to accommodate peak loading on the new switching system. Results have been scaled proportionately for support tasks like product and project management using similar percentages.

TABLE 4-5 Custom vs. COTS and open-source software development comparison.

Task	Custom Software Development		COTS and Open-Source Package Usage	
	Schedule (months)[a]	Effort (in millions)[b]	Schedule (months)[c]	Effort (in millions)[d]
Systems engineering	6	$ 25[e]	6	$ 30
Project management	Level of effort	75	Level of effort	45
Product support	Level of effort	55	Level of effort	30
Software engineering	66	370	24	201
System integration and test	12	55	12	30
Systems test	3	10	3	10
Deployment	3	5	3	5
TOTALS	**72**	**$595**	**36**	**$351**

[a] Tasks will be done in parallel to achieve delivery in 3 years.

[b] The shortest feasible schedule according to COCOMO II model as calibrated is 6 years.

[c] Effort was computed using a nominal profile for telecommunications and a size base of 4 MSLOC (million source lines of code). Cost/staff-year of effort is assumed $200,000.

[d] Numbers were taken from Table 4-1 in this chapter and scaled to include software costs only.

[e] COCOMO II model estimates the cost is less than the estimates submitted by the engineering staff.

Outcomes and lessons learned

The numbers in Table 4-5 tell the story. Developing custom software doubles the schedule and adds $244 million, or about a 70 percent increase in costs, to the effort. Yes, there might be risk in using COTS and open-source software. But there is lots of risk in pursuing a custom development as well.[5] Also, the desired three-year schedule for the project is infeasible if you use custom development.

When you review your numbers with the planning project team, they are impressed. But your team lead is not. He says the battle has just started and that you better check and double-check your numbers. He suggests that the engineering department will use every trick in the book to discredit you. Power, jobs, and budget are on the line. Therefore, the fighting will be fierce. He asks the member of your team from the finance group to dig up past engineering costs to determine if the cost model's assumptions are in line with actual results. He then asks the team if they know of any metrics that senior management uses to test estimates. The response is that the magic number used in the firm for validation is $100 per SLOC (source line of code). Where that figure comes from nobody knows. However, when you apply it, the results of using this rule of thumb seem to compare nicely with the $92.50 per SLOC that was the output of the cost model ($370 million divided by 400 MSLOC).

The financial person on the team reports his findings, which are surprising. The cost per SLOC actually delivered six years ago for the switching system that is currently in the field was $125 per SLOC. When you think about it, you can build a case that you would expect current costs to be less, especially because you plan to exploit advances that have been made in technology when you build the new system. You and your boss feel comfortable about your numbers and feel ready to defend them when management calls you as they try to settle the debate over the use of COTS and open source. In the interim, you plan to research the numbers more completely in an attempt to further validate your findings. Your strategy is to let the numbers do the talking.[6] Management can ignore them if they want, but they paint a compelling case for the use of COTS and open-source software packages in the new switching system. As a change agent, you also need to recognize the need to use techniques to address resistance.

The lessons learned in this telecommunications case were many and include, but are not limited to, the following:

- COTS and open-source software represent new ways of doing business. As such, you should apply change management principles to enable early, frequent, and ongoing communications with stakeholders to deal with the inevitable resistance that will be encountered.
- There are many advantages and disadvantages associated with the use of COTS and open-source packages. Be sure to evaluate each carefully before making a commitment.
- COTS and open-source software packages do not come for free. Besides the license costs, there are other expenses associated with getting the package ready to be interfaced and used as part of the system in which it will operate.

- When looking to use COTS and open-source packages in systems, be sure to identify and license all of the packages you need in the systems, applications, and support domains during development, maintenance, and operations (run-time licenses).
- Recognize that most COTS and open-source packages do not plug and play directly out of the box. Some effort might be needed to configure, tailor, integrate, maintain, and sustain these packages throughout the life cycle. This is especially true if there are architectural mismatches and the packages do not support your data model.
- Maintenance of COTS and open-source software packages can be difficult because they are updated at a different frequency than the system they are part of. In response, you have to plan to synchronize package updates with your releases and map the features.
- Plan also to try to influence the direction COTS and open-source software package vendors take through relationship management. Although you might not be the vendor's biggest customer, you want to be one of their most important ones. Realize that you might have to pay a premium or make investments to achieve this status.
- When performing a make/buy analysis as in the case study, recognize that the numbers will do the talking unless there are compelling reasons why they should be discarded.
- Take care to make your numbers believable and credible. Whenever possible, validate them against your past performance and rules of thumb that are part of your firm's history.

Summary

This chapter provides those who are planning projects with insights into how to spot, quantify, and deal with controversial issues. Controversy in planning almost always revolves around risk. "Risk" in this sense is defined in terms of exposure to adverse effects, schedule delays, or cost excesses. Because risk can make change more difficult, it should be considered carefully. Once risk is quantified, the trick in risk management is to let the numbers do the talking. I have seldom seen management select the most costly option. In those rare situations where I have, there has been some urgent business reason like keeping an operation afloat until a new contract is negotiated or a takeover attempt has been consummated. In this case study, the really risky and difficult issues associated with COTS selection, tailoring, integration, and sustainability were not even brought to the table. The reason for this was simple. If I had raised these issues, this chapter would have taken too much space and still might not have provided a proper treatment. However, be careful with evaluating COTS solutions, and realize that they need to be handled carefully. When and if you do use COTS and open-source solutions, you'll see why I provided you with a warning.

References

References cited within this chapter include the following:

[1] Kurt C. Wallnau, Scott A. Hissam, and Robert C. Seacord, *Building Systems from Commercial Components* (Addison-Wesley, 2002).

[2] Scott Hissam, Charles B. Weinstock, Daniel Plakosh, and Jai Asundi, *Perspectives on Open Source Software,* Software Engineering Institute, Report No. CMU/SEI-2001-TR-019, November 2001.

[3] Paul Clements and Linda Northrop, *Software Product Lines* (Addison-Wesley, 2002).

[4] Barry W. Boehm, Chris Abts, Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald Reifer, and Bert Steece, *Software Cost Estimation with COCOMO II* (Prentice Hall, 2000).

[5] C. Ravindranath Pandian, *Applied Software Risk Management: A Guide for Software Project Managers* (Auerbach Publications, 2006).

[6] Donald J. Reifer, *Making the Software Business Case: Improvement by the Numbers* (Addison-Wesley, 2001).

Web resources

Applicable web resources that amplify points made in this chapter can be found here:

- Amazon has many books and articles on subjects discussed in this chapter. Go to *www.amazon.com* and search under the headings of *COTS*, *open source*, *product lines*, and *risk management* to find relevant citations.
- The Software Engineering Institute (SEI) at *www.sei.cmu.edu/library/* also has many citations on these subject areas because they happen to be an interest area of their primary customers.
- A briefing by Dorothy McKinney about the pluses and minuses of COTS use is available at *http://www.incose.org/northstar/2001Slides/McKinney%20Charts.pdf*.
- An interesting case study about COTS software has been prepared by the faculty at McGill University and is available at the following website: *http://www.cais-acsi.ca/proceedings/2000/beheshti_2000.pdf*.
- A COTS Risk Mitigation Guide developed by the Federal Aviation Administration in 2010 is available at *http://fast.faa.gov/docs/COTS%20Risk%203.1a%20Guide.doc*.
- A briefing article about COTS security vulnerabilities by Craig Miller of Cigital is available at *https://buildsecurityin.us-cert.gov/bsi/623-BSI.html?branch=1&language=1*.
- Another briefing about COTS hardware and software for NASA's Earth Observing System can be viewed at *http://edhs1.gsfc.nasa.gov/waisdata/rel6/cd71560001_4.pdf*.

- Information about the Open Source Initiative (OSI), which is a nonprofit corporation whose mission is to educate and advocate for the use of open-source software, is available at *http://www.opensource.org*.
- A comprehensive manual describing how to develop open-source software for Intel's PCI Express family of gigabit Ethernet controllers can be viewed at *http://download.intel.com/design/network/manuals/316080.pdf*.

CHAPTER 6

Industrial Case: Utility Moving to the Clouds

Setting the stage

The utility company you work for is considering moving to cloud computing. Cloud computing, as you find out via the Internet, is a relatively new concept that refers to an assortment of logical computational resources that are made available via computer networks rather than on local computers.[1,2] Applications are hosted on multiple servers across the cloud. Data is also stored on server farms. In this manner, both applications and data can be made accessible via a browser rather than you having to install and run them on your desktop, laptop, or office server. Instead, both run on the cloud via its servers and results are made available through the network to clients on their computers. Clouds can be public and private, depending on the need. In addition, cloud services are sold on a demand basis using any of these three arrangements:

- **Software as a service** End-user applications services are accessed over the network rather than on client computers. Under this arrangement, you execute your business application remotely to get results typically at a fraction of the cost of licensing the software.
- **Platform as a service** Sets of application components can be put together by developers via plug-and-play and run on cloud-computing servers to get results.
- **Infrastructure as a service** Developers can build applications from scratch and run them in virtual machines on the cloud servers without having to license tooling that can be costly.

The major advantage of cloud computing is its significantly lower cost relative to the older model, where you would acquire and maintain hardware and software resources. It removes the need for large capital investments in equipment, infrastructure, and software and reduces related operating costs proportionately. It also increases potential mobility because the only thing workers need to do, wherever they are, to access computational resources is connect to the cloud.

The disadvantages of cloud computing are many and include becoming dependent on someone else to control your computational resources. As a consequence, you will fail if they do or if they are unwilling to pitch in to resolve a crisis. Besides other disadvantages, the cloud has serious security and privacy risks, especially if confidential data is not protected adequately. Obviously, there is lots of information available about cloud computing in the professional literature[3,4] and on the Internet. (See the end of this chapter for pointers to these resources.)

Your task in this case study is to lead an internal team that has been asked by management to determine whether or not to use cloud computing to provide basic services for your firm and its customers. Customers are residential, industrial, and governmental users of the gas, power, energy, water, and waste-removal services that your firm offers on a fee-for-service basis throughout the municipality.

Organization

Figure 6-1 shows an organizational chart of the entire utility company.

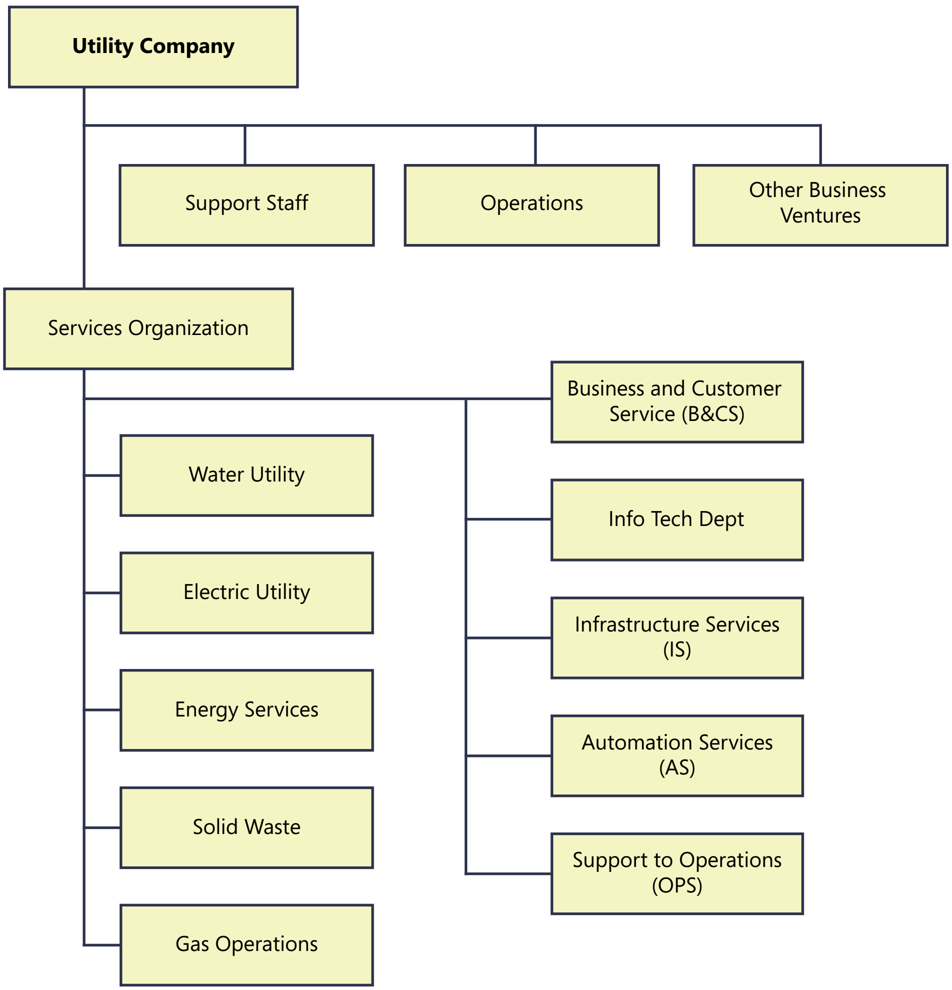


FIGURE 6-1 Organizational chart for the utility company.

The part of the company you are most interested in is the Services organization, which provides a wide range of utility services for a large city and its suburbs. The Information Technology (IT) department of which you are a part provides centralized support for each of these groups from the company's main offices located in a large city in the southern United States. The department provides infrastructure services for the enterprise as a whole, and automation and operations support services for each of the groups that are part of the Services organization. In addition, IT has a major field site co-located with the services group's Business and Customer Service (B&CS) organization, which is about 30 miles away. This field site was built as a backup for the main site in case a disaster occurs. Both sites have development facilities and service the firm's client/server networks. The IT department has 1,200 professionals located at these sites and 300 support personnel. The IT general manager also serves as the chief information officer (CIO) of the company. However, the vice president of operations is responsible for ensuring that the services provided by operations personnel is exemplary at the many sites the company maintains across the country in cities, counties, and government facilities.

Project

The project you are in charge of is primarily tasked with determining whether or not to use cloud computing within the IT department to lower operating costs for infrastructure services like payroll and travel. You have a team of three who are working with you part time to develop a recommendation regarding the use of what management views as a beneficial, cost-cutting technology.

During your team's kickoff meeting, the IT general manager provides the following added information and direction for the effort:

- Assess whether moving to cloud computing makes sense for B&CS and Infrastructure Services (IS) because they each maintain various resources that other groups access via the company's client-server network. The capital assets of each group (equipment, licenses, and other items) are listed in Table 6-1.
- The number of staff currently required to operate and maintain B&CS and IS facilities and equipment is listed in Table 6-2. These counts address common software used across the company. They do not include the staff used to develop new software applications, who are funded separately by the operational groups.
- Determine the impact of cloud computing on the IT department's capital costs of doing business. As part of this analysis, look at what happens to the equipment and facilities that will be disposed of when the department transitions to the use of cloud computing.
- Make sure that no matter what you do the same quality of service (or better service) is provided to your customers. This is determined by using the current primary measure, which assesses response time relative to a customer's request for service.
- Retain any applications that provide the company with clear advantages over the competition, such as the company's current ability to read meters remotely to tabulate billings.

- Investigate how to address the security and privacy shortfalls of cloud computing. Make sure customer proprietary information is protected regardless of the cost.

TABLE 6-1 Capital assessment for the B&CS and IS groups.

Organization	Asset	Purchase Price (in millions)	Accumulated Depreciation[b] (in millions)	Current Book Value (in millions)
B&CS	Computer equipment	$40	$20	$20
	Communications gear	15	8	7
	Software licenses	20	7	13
	Capital improvements[a]	25	10	15
	TOTALS	**$100**	**$45**	**$55**
IS	Computer equipment	$60	$25	$35
	Communications gear	20	10	10
	Software licenses	20	8	12
	Capital improvements[a]	50	22	28
	TOTALS	**$150**	**$65**	**$85**

[a] Expenses to improve leased facilities (long term)

[b] Using IRS guidelines for depreciation for different types of equipment and licenses

TABLE 6-2 Annual average staff expenses to maintain and operate computational resources.

Organization	Task	Annual Cost (in millions)[a]
B&CS	Hardware and software maintenance	$2
	Sustaining engineering, including user support	2
	Operational support (includes call center)	4
	Product management	1
	TOTALS	$9 (45 people)
IS	Hardware and software maintenance	$5
	Sustaining engineering, including user support	2
	Operational support	2
	Product management	1
	TOTALS	$10 (50 people)

[a] Cost for a staff-year of effort is assumed to be $200,000 at this price level.

Process

The approach your team decides to take to make your determinations and findings adheres to the practice your firm has in place for such analyses. The steps are outlined as follows:

- **Step 1: Develop a Concept Paper** Summarize your concept of operations in a white paper for cloud computing use in B&CS, and summarize its use in IS in a white paper that highlights the capabilities you hope to acquire from the vendors.
- **Step 2: Issue a Public Request for Information** Issue a public Request for Information (RFI) asking vendors to comment on your white paper and tell you how they would go about satisfying its requirements.
- **Step 3: Gather Information/Develop Requirements** Hold discussions with vendors who provide technically acceptable responses to the list of items you asked them to address in your white paper. These include the evaluation criteria you identified in the paper to be used to make such determinations and findings.
- **Step 4: Prepare/Issue a Request for Proposal (RFP)** Issue an RFP to vendors who responded to your RFI for the acquisition of cloud-computing systems and services based on a solicitation that contains your requirements, work statement, and preferred contractual terms and conditions. Make sure that your solicitation does not bias the acquisition by using vendor proprietary information gathered via the RFI process. Otherwise, you might have to deal with a protest from one of the losing vendors.
- **Step 5: Conduct Source Evaluation/Selection** Using criteria (responsive to requirements, lowest cost, minimum risk, and other such items) contained within the solicitation, rate and rank the vendor proposals. Make a selection based on the given criteria. Identify major strengths and weaknesses in the winning proposal, and forward it along with your recommendations to those responsible for negotiating a contract.
- **Step 6: Issue a Contract** Negotiate with the selected vendor to acquire the products and services using best value and fairness as your overriding principles. Work with the vendor to take advantage of its strengths and compensate for its weaknesses. Remember, the vendor must succeed for you to succeed.
- **Step 7: Monitor the Contractor/Accept Delivery** Provide oversight and direction as the contractor works to comply with your requirements for delivery of acceptable products and services. Accept delivery only when the vendor supplies evidence of compliance with your contract requirements.
- **Step 8: Commence Operations** Using your concept of operations, transition to the use of the cloud-computing products and services acquired in as disciplined, logical, and risk-free a manner as possible. Apply change-management principles during the transition. Remember to plan in detail because the transition might require you to operate systems in parallel to minimize potential impacts when running a shop 24 hours a day/7 days a week. Be sure to include recurring activities, such as maintaining ongoing communications with stakeholders.

The response to the RFI is overwhelming. You received 42 replies, of which 18 vendors seem to satisfy your requirements. In addition, the telephone has not been idle for the past two days. Most of these phone calls are vendors asking for time to visit and present their wares to you and your team. However, your timeline will not accommodate them all. You need to cut the number to six vendors, at most. You do this by asking all of those responding if they have experience with utility companies similar to yours. Because only three of the six can respond positively, you can reduce the list of promising vendors accordingly.

Product

After discussions with the vendors, you feel that you have the information you need to pull together a briefing to your boss and his staff on how to exploit the use of cloud computing within your utility company. Your briefing will contain the following observations and recommendations relative to the changeover:

- The move to private cloud computing, where facilities are dedicated to the company, has many benefits as confirmed by government studies[5,6] and all of the vendor sales pitches. Costs can be substantially reduced, and the company's ability to expand and contract its computing resources as needed is greatly enhanced.
- Given the current business picture and constrained B&CS and IS budgets, private cloud computing seems to represent a viable path forward for the utility.
- The ability to implement a measured service under a pay-for-use paradigm, which provides services on an on-demand basis across a ubiquitous network, has many advantages.
- All those interviewed concurred that private cloud computing seems to make the most sense for IS because the services it provides are for the entire organization. Because B&CS is localized, it does not seem to make sense to use private clouds for them on a broader basis.
- Cost savings will be realized as a function of the substitution of virtualized applications software in the cloud for labor and facilities from current dedicated resources.
- Cost savings from the cloud can be realized in stages as various applications are replaced by vendor replacements. Transition to private cloud computing will outsource general business applications first and then operations and maintenance later.
- The team recommends going forward with private clouds for IS but not for B&CS. The next steps in the process will pull together the requirements, develop a solicitation, and issue an RFP for the acquisition of products and services.
- In preparing for the RFP, the team will identify and seek to retain core services that are fundamental to the way the utility does business and that represent a competitive advantage, such as the ability to read meters remotely and bill clients directly for services, as mentioned earlier.
- In the RFP, current systems will be kept operational and working in parallel during a transition period of three to five years. Some business process reengineering will be required during the transition to optimize how the new cloud-computing resources are used.

- In the RFP, as a risk mitigation action, those proposing solutions will be asked how they can address known weaknesses of cloud computing, such as those related to security and privacy.
- In the RFP, purchase and maintenance options will be included to provide the utility with leverage once the acquisition is completed and the vendor is under contract.

People

Your boss liked your briefing, nodded his head in agreement many times as he listened, and concurred when you concluded with all of your team's recommendations. However, the head of IS was infuriated and vocally criticized every one of your charts. Such protests were expected because IS would be taking the brunt of the cuts. When you were asked how much could be saved, you replied, "Based on vendor inputs during the question and answer sessions, they estimated savings between $50 and $60 million of equipment at book value and from 35 to 40 people with a total staff cost between $7 and $8 million. The total reduction based on these numbers is between $57 and $68 million from the current budget of $95 million."

The head of IS immediately responds to these numbers with a blistering rebuttal. He states that such savings are unrealistic because much of this equipment and the people will have to be retained to run existing facilities in parallel during the three-to-five-year transition period. In addition, at least five new people will have to be hired during this period to perform the business process reengineering tasks, including the staff needed to train users in their proper utilization. He estimates that the conversion costs during the three to five years will add $8 to $10 million to IS's current operating expenses. These numbers rattle you, the general manager, and the audience. "He is right," says the general manager. "You need to investigate the costs of transition more fully before I make a go/no-go decision on the acquisition," he continues.

The head of IS has a grin on his face and looks pleased. He volunteers to have two of his best senior people work with you part time as you develop a response. You politely decline, but the general manager thinks it is a good idea, and you reluctantly accept the offer of help. The general manager schedules an additional meeting two weeks from this one to review the cloud-computing recommendations again.

You should have expected and prepared for the IS response because anticipating and planning for resistance is a fundamental change-management principle. But you did not. Getting cloud computing accepted now will be harder. But it still seems doable.

Options, recommendations, and reactions

The team gets together to assess the options with the transition in mind. They identify the following four main transition scenarios to cloud computing that everyone agrees make sense:

- **Option 1: General Application-Only Transition** Transition most general applications to a private cloud, retain IS facilities and staff to run general applications, and continue servicing customers on a fee-for-service basis.

- **Option 2: Partial Facilities Transition + Option 1** Perform Option 1, and shut down unneeded facilities within IS. Sell off equipment, and reduce staff proportionately as private cloud services and applications become operational. Upgrade equipment as needed to address reliability issues.
- **Option 3: Transition to Upgraded Facilities + Fuller Set of Applications** During the transition to the private cloud, upgrade facilities to provide core processing and backup. Address current equipment reliability issues that are occurring as gear ages and failures increase, thus jeopardizing 24/7 operations. Sell off equipment, and adjust staff proportionately as facilities and cloud services and applications become operational.
- **Option 4: Operate IS As-Is** Upgrade IS equipment to address reliability problems, and continue to operate as-is. Perform some streamlining to cut costs and improve service to consumers.

Table 6-3 summarizes the results of the team's analyses after considerable debate. It identifies the major strengths and weaknesses of each option, along with the estimated costs and projected benefits. The table seems to highlight the overall conclusion that movement to one of the three private cloud-computing options is the right thing to do even though IS remains reluctant to support such a recommendation (that is, the two people assigned to your team neither concurred with this analysis nor agreed to put their names on the results).

TABLE 6-3 Strengths and weaknesses of cloud-computing options.

Options	Strength	Weakness	Cost	Benefits
General Applications Only in Clouds	■ Big gains, little effort ■ Can handle increased workloads ■ Minimal disturbance	■ Does not address the reliability issue	■ Transition cost of $25 million ■ Two to three years before benefits accrue	■ Cost avoidance of $40 million a year ■ Increased flexibility ■ Can handle a larger workload
Partial Facilities + Partial Applications in Clouds	■ Improve ability to respond quickly as business conditions change via the cloud ■ Addresses reliability issues	■ Transition is hard and takes time ■ New jobs must be found for displaced staff ■ Business processes must be updated	■ Transition cost of $50 million ■ Three to five years before benefits accrue	■ Cost avoidance of $50 million a year ■ Better business processes ■ Reliability issues handled ■ Less equipment and staff to worry about
Upgrade Facilities + Fuller Set of Applications in Clouds	■ Same strengths as Option 2 plus company can back up clouds with its own facilities	■ Same weaknesses of Option 2 plus more turmoil during transition	■ Transition cost of $65 million ■ Three to five years before benefits accrue	■ Cost avoidance of $65 million a year ■ All the benefits listed previously
Operate IS As-Is	■ Minimum pain	■ Minimum gain	■ Minimum cost	■ Reliability issue is addressed

Outcomes and lessons learned

Your next meeting with the IT general manager and the head of Infrastructure Services is stormy. It is apparent during the meeting that the head of IS is unhappy with the results. He continuously bombards you during your briefing with nasty remarks, and he accuses you several times of deliberately ignoring his people's inputs. In addition, he blasts the legitimacy of the numbers and asks for details on how each was derived. You respond with the spreadsheets that provide backup and tell him that his own people reviewed the numbers and found them reasonable.

Your boss finally has no option but to tell everyone to cool down. He states that even though the numbers speak for themselves and seem to present a solid business case for change, he has concerns. His major trepidations, he says, are the risks associated with the transition and the displacement of personnel. He says that equipment has to be changed no matter what option is chosen because it is wearing out and the reliability declines have to be fixed. Based on his remarks, it is not surprising that he accepts Option 2. In response, you and your team take the action to move to the next step in the process by preparing and issuing an RFP. You hope that several of the vendors who replied to your RFI will respond to your RFP with proposals that provide good value for your money.

Your team meets and tries to scope what goes into the RFP besides the requirements and boilerplate text. A member of your team who has been through a large purchase like this before advises you to pay attention to the boilerplate text because this is where the evaluation criteria for selection and the terms and conditions for the purchase go. That's good advice, you think. So you schedule a meeting between your team and the Purchasing staff.

The meeting with the Purchasing staff goes very well. They had lots of experience and advice about what to put in the solicitation document. Key provisions include rewards for delivering early and penalties for being late. They also provide options to acquire several products and services (additional applications and services, more equipment, training, and other such items) that can be taken after the contract is awarded at a fixed price. Maintenance terms and conditions for the first five years of operations were also spelled out so that you can get the vendor's immediate and undivided attention when problems occur after delivery.

At the suggestion of the Purchasing department, you send the solicitation out to the prospective suppliers for comment prior to releasing it. You get back a lot of constructive criticism and suggestions. You find that the most controversial clauses are those associated with late delivery and maintenance.

The lessons learned in this industrial case study were many and include, but are not limited to, the following:

- Even when you think there is a clear choice, resistance to change can pop up from unexpected sources. Therefore, also anticipate resistance and plan to deal with it.
- Resistance to change comes primarily from those whose power, staff, and budgets are threatened. In this case, such cutbacks are real threats to the Infrastructure Services group.

- Those who foster change need to anticipate the perceived threats and develop plans to help address them as part of their effort. In this case, figuring out how to find other positions for staff who are no longer needed might have alleviated some of the pain.
- There might be hidden issues that influence decisions relative to change. In this case, aging equipment and reliability issues did not surface until late in the process when options were being compared. Yet, the issue was one of the major drivers in determining which option was selected.
- Using competitive market forces to seek the best alternative can be beneficial, especially if you can get an expert review of your solicitation by stimulating the vendors to provide you inputs as to which of your requirements are feasible and which are not.
- However, relying solely on vendor inputs is dangerous. Because they want to make a sale, they might stretch facts and cloud reality by confusing current capabilities with future capabilities.
- Using strengths and weaknesses along with costs and benefits permits stronger cases to be made for recommended alternatives.
- Getting a vendor on contract takes considerable time and effort. It also forces you to solidify your concepts of operations, requirements, and contract terms and conditions.
- Getting selected vendors to deliver what they promise often takes patience, effort, and due diligence. Many will do a good job. Others may let you down after the contract is issued. To succeed, plan to manage rather than monitor the contract. Otherwise, you probably will get less than what you expect and less than what you are paying for.
- The challenge will occur after the cloud products and services are delivered and accepted. If you are not one of the vendor's key accounts, keeping their attention during operations and maintenance might become an issue. That is why I strongly recommend negotiating terms and conditions for any follow-on maintenance contract as part of the original acquisition.

Summary

This chapter provides insight into large procurements for Information Technology (IT) products and services. The major issues in this case revolve around addressing resistance to change brought on by a changeover to a new computing paradigm—for example, cloud computing. In this case, such resistance should have been anticipated and dealt with earlier in the process. The team should have gotten IS personnel involved earlier and solicited their inputs and resolved their objections prior to making the recommendation for one of the change options. In the process, they would have learned about and been able to attack the issues of reliability and placement of staff. Instead, they became involved in a war of words that detracted from the goal of the effort, which was determining whether or not cloud computing made sense for this utility company.

This is another chapter where I cut back on materials to save space and maintain a focus. Please understand that cloud computing is controversial and has many issues associated with it that deserve further coverage. For example, the performance of the cloud is overstated—that is, the vendor often promises more performance at a lower cost than it can deliver. As another example, the tools you use may or may not work as advertised on the cloud. Be warned that you need to move carefully to the clouds because they are still in their early-adopter period.

References

References cited within this chapter include the following:

[1] Charles Babcock, "Why Private Cloud Computing Is Real—And Worth Considering," *InformationWeek*, April 11, 2009.

[2] Amazon web services provides numerous case studies on cloud computing at the following site: *http://aws.amazon.com/solutions/case-studies*.

[3] The Cloud Computing Journal has published numerous success stories that can be reviewed at *http://cloudcomputing.sys-con.com/node/1687873*.

[4] Marc Benioff and Carlye Adler, *Behind the Cloud: The Untold Story of How Salesforce.com Went from Idea to Billion-Dollar Company and Revolutionized an Industry* (Jossey-Bass, 2009).

[5] Government Computer News devoted an entire issue of the magazine in 2011 to cloud computing successes that is accessible at *http://gcn.com/microsites/2011/cloud-computing-snapshot/index.aspx*.

[6] NASA Nebula Project. Information about their use of cloud computing is available at the following site: *http://nebula.nasa.gov*.

Web resources

Applicable web resources that amplify points made in this chapter can be found here:

- Amazon has many books and articles on the subject of cloud computing. Go to *http://www.amazon.com* to find relevant publications.
- Amazon also provides infrastructure web services via a cloud platform for commercial companies of all sizes via a fee-for-service arrangement for application hosting, backup and storage, e-commerce, media hosting, search engines, web hosting, and other services.
- YouTube has a number of videos on cloud computing, like the following one that introduces you to the technology: *http://www.youtube.com/watch?v=ae_DKNwK_ms*.
- Many technology firms in the cloud-computing business provide resources to help those investigating this technology. For example, Hewlett-Packard offers a planning guide at its cloud-computing digital hub at *http://www.techlearning.com/article.aspx?Id=39248*.
- The Open Cloud Consortium (OCC) manages cloud computing resources and develops reference implementations, benchmarks, and standards for its members that support scientific

research. Information about the consortium is available at *http://opencloudconsortium.org/about*.

- IBM and the European Union have announced the formation of a consortium to research new computing models. See *http://www-03.ibm.com/press/us/en/pressrelease/33067.wss*.
- The Cloud Computing Journal offers lots of articles on the use of the technology at *http://cloudcomputing.sys-con.com*.
- Another useful publication called the Cloudbook, located at the following site, provides relevant articles: *http://www.cloudbook.net*.
- Lots of technology firms, training companies, and universities offer training in cloud-computing areas.
- Major conferences and expositions on cloud computing such as the following are held in large cities worldwide: *http://www.interop.com/newyork/conference/cloud-computing.php*.
- A current list of cloud-computing discussion groups, list servers, and blogs can be found at *http://www.thedacs.com/databases/url/key/7848/7852*.
- A nonprofit group of major technology firms has been formed to address cloud-computing security issues. It is called the Cloud Security Alliance, and it has many resources at the following site: *https://cloudsecurityalliance.org*.

Index

D

E

F

G

H

I

J

K

L

M

Q

R

U

V

W

About the Author

DONALD J. REIFER is one of the leading figures in the field of systems/software engineering and management, with over 40 years of progressive experience. He has built businesses, steered troubled projects, and served in executive positions in industry and government. He specializes in the area of metrics and measurement. Besides being a trusted advisor to major corporations and the government, he has founded software businesses and served as an expert witness.

From 1993 to 1995, Mr. Reifer managed the DoD Software Initiatives Office under an Intergovernmental Personnel Act assignment with the Defense Information Systems Agency (DISA). As part of this Senior Executive Service (Three Star Equivalent) assignment, he also served as the Director of the DoD Software Reuse Initiative and Chief of the Ada Joint Program Office and worked for the Office of the Assistant Secretary of Defense (ASD/C3I), who served as the CIO for the Department of Defense.

Previously, while with TRW Inc., Mr. Reifer served as Deputy Program Manager for their Global Positioning System (GPS) efforts. While with the Aerospace Corporation, Mr. Reifer managed all of the software efforts related to the Space Transportation System (Space Shuttle). Mr. Reifer started his software career at Hughes Aircraft Company as a software engineer on the Phoenix missile project.

Currently, as President of Reifer Consultants, Mr. Reifer supports executives in many Fortune 500 firms who are developing investment strategies aimed at improving their systems and software engineering capabilities and capacity. Mr. Reifer also serves as the Principal Investigator on the firm's information warfare and software protection Small Business Innovation Research program (SBIR) efforts. For the past three years, he has been leading an Army study looking at ways to improve weapons system maintenance and sustaining engineering support. He also serves as a Visiting Associate at the Center for Systems and Software Engineering at the University of Southern California (USC) where he is involved in the COCOMO project.

Mr. Reifer was awarded the Secretary of Defense's Medal for Outstanding Public Service in 1995 for the innovations he brought to the DoD during his assignment. Some of his many other honors include the AIAA Software Engineering Award, Hughes Aircraft Company Fellowship, the Frieman Award for advancing the field of parametrics (cost estimation), and membership in Who's Who in the West. Reifer is a senior member of the ACM and IEEE and a member of INCOSE and ISPA.