PRENTICE HALL

APPLYING DESIGN FOR SIX SIGMA TO SOFTWARE AND HARDWARE SYSTEMS

Eric Maass | Patricia D. McNair

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/ph

Library of Congress Cataloging-in-Publication Data
Maass, Eric.
Applying design for six sigma to software and hardware systems/Eric Maass, Patricia D. McNair.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-714430-X (hardback : alk. paper) 1. New products. 2. Six sigma (Quality control standard) I. McNair, Patricia D. II. Title.

TS170.M33 2009
658.5'75—dc22 2009021446

ISBN-13: 978-0-13-714430-3
ISBN-10: 0-13-714430-X
Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.
First printing, August 2009

Foreword

The challenge of developing and launching a successful product into the marketplace is dependent on the effective resolution of a series of compromises: compromises between design and iteration, research and execution, development and testing, and so on. The ability to quickly and accurately work one's way through this process often spells the difference between a product that is successful in the market and one that is not. The emergence and availability of tools and techniques that can inform these decisions and help improve the "hit rate" of success therefore becomes more and more important.

Product development can be summarized as the process of answering two fairly simple questions: "What is it?" and "How can we tell when we are done?" The ability to clearly and objectively address these questions under significant time and resource pressures distinguishes the top product operations from others.

As one evaluates successes and failures in the product space, it seems that some products have almost a unique "voice" of their own. Whether a phenomenon like the original Motorola RAZR phone, the revolution-causing Apple iPod MP3 player, or the digital video recorder, these industry-changing products are unique in that they specifically address unmet needs. It is notable that only a very few of the actual features of these products form the basis for their success; the Apple iPod wasn't particularly great in audio quality, the RAZR had less talk time than most competitive offerings, but in both cases the excellence and targeting of the anchor attributes outweighed the more minor shortcomings. I once heard a very senior colleague of mine state, without fear of contradiction, that there are no examples of great products that are purely the result of

consumer or end-user research. The gist of this comment is that consumers haven't encountered all of the unmet needs that distinguish truly innovative products. This would lead to the need for techniques that integrate the consumer insight process with the potentials for applicable technical innovation in the space. While there is no panacea to this need, the ability to use objective techniques in this space is fundamental to success, particularly in deciding where to focus the time, resources, and costs of the product to get maximum leverage in the marketplace.

The concept of "cost of quality" in its most extended state is a very powerful metaphor for the effectiveness of a development cycle. Simply stated, it is the allocation of all effort into two categories—"value added" and "defect detection and extraction"—and the use of proactive tools and techniques to increase the first at the expense of the second. Let me elaborate. If we hypothesize a perfect product development cycle—crystal clear definition optimally tied to the user target, rendering of this description into the relevant software and hardware sub-elements, and then flawless execution without the introduction of any defects—we arrive at the irreducible minimum cost and time for a product development cycle. Great organizations take on the challenge of identifying their current level of performance, comparing it to their competitors, and then setting out to reduce this cost of error by 15% to 20% per year, using clearly defined and communicated tools, methods, and technology improvements.

The third important factor in this discussion is the organizational or human element: how does one deploy new techniques and approaches in a mature organization, overcoming the "not invented here" tendencies of all engineering professionals, and quickly traverse the learning curve phase to achieve results? Here is where the deployment and institutionalization aspects developed in Six Sigma and extended for Design for Six Sigma (DFSS) bring significant value. The combination of formal training, implementation of highly experienced mentors into actual development projects, and gradual development of a "community of practice" has been found to be an extremely effective approach.

Making great products is a combination of art and science, the art being the use of experience, insight, and intuition to decide how much of the available science to employ. DFSS is a structured method for developing new products that are aligned with the customers' needs, using predictive engineering and anticipating and managing potential issues. The authors of this book have developed a unique concept for applying DFSS to hardware and software systems. The collected series of methods, tools, and techniques has been proven in application in leading organizations in a variety of industries. While there is no such thing as a definitive product design "cookbook" that infallibly results in industry leading products, this volume represents a rich collection of techniques and approaches that, if used properly, can identify and address the "sweet

spot" aspects of a product definition, proactively identify the high leverage realization challenges, and predict and resolve issues early in the process. The combination of these techniques with talented and trained facilitators in Six Sigma methodologies and change management approaches can and will have major impact in both the effectiveness and efficiency of any product development organization.

—Rey Moré
Former Senior Vice President and Chief Quality Officer
Motorola, Inc.

Preface

Purpose and Scope

The goal of this book is to provide a clear roadmap and guidance for developing products—not only simple products but also high-tech, information age products and systems involving both software and hardware development. The intent is to provide clear, practical guidance with real and realistic examples so that the reader will have exactly what he or she needs to successfully apply Design for Six Sigma (DFSS) to products and system development projects involving software, hardware, or both.

The scope of the book encompasses the development project from the development and justification or prioritization of the business case and the associated project schedule through the developing of customer-driven requirements and consequent efforts to fulfill those requirements with high confidence.

DFSS is a structured method for developing robust new products that are aligned with the voice of the customer (VOC), using predictive engineering and anticipating and managing potential issues. Using this proactive process, the development team can:

- Ensure that the team shares a solid understanding of customer needs, and selects a compelling concept that supports and facilitates meeting those needs.
- Define measurable critical parameters that reflect customer needs, and flow them down to quantifiable and verifiable requirements for the subprocesses, subsystems, and components.
- Use predictive engineering and advanced optimization methods to ensure that the product, technology, service, or process is robust to variation in the processing and in

the environment and use conditions, providing confidence that each critical parameter will meet or exceed customer expectations.
- Verify that the new product, technology, and service of process is capable of fulfilling the functional and reliability requirements under normal and stressful conditions.
- Ensure that the supportive organizations and supply chain are aligned and capable of consistently delivering with acceptable cycle times.

Who Can Benefit from this Book

Although this book is general in approach and could be very helpful for anyone involved in developing almost any product, this book is particularly attuned to meeting the needs of highly skilled people who are motivated to take their new product efforts to the next level.

This book provides the tools and step-by-step guidance for systems engineers, programmers, software engineers, electrical engineers, engineering managers, program managers, and engineering students. It will also be useful for engineers who handle multidisciplinary situations involving software or electronic hardware, such as aerospace, biomedical, and industrial and power engineering. Software and electronics has seeped into many other disciplines as well, so it will also be useful for mechanical engineers, chemical engineers, and civil engineers.

For perhaps the first time, skilled people involved in product development have access to a clear roadmap, clear guidance on what people and teams need to do, step by step, to apply powerful methods—sometimes simple yet elegant, other times more complex—that can enable the product development team to converge quickly on excellent approaches and solutions to deal with even the most complex situations with high-tech and information-age products.

This book addresses a common concern from people who read books and then wonder, "That's great in theory—but what do I need to do?"

Many products involve both software and hardware aspects; this book provides an integrated systems approach that pulls the marketing, software, and hardware communities together to solve hardware and software issues and provide a product that meets customers' expectations, with minimal finger-pointing.

Organization and Summary of the Chapters

This book is organized in approximately the same sequence in which the topics are most likely to arise for a new product launch. Although it is hoped that readers will find

the engaging literary style grabs them like a fast-paced novel, such that they would read it through in one spellbound sitting, the reader can simply read each topic just as the need arises in the actual product development effort.

The first three chapters set the context and provide the reader with the background and a structure to assist in the challenges involved in Six Sigma deployment in general and DFSS deployment in particular. The first chapter provides a historical perspective followed by a summary of the DFSS process, goals, and an example of a DFSS project. The second chapter provides the deployment perspective, and gives information and tools to assist with the organizational and people challenges involved in DFSS deployment—approaches for engaging management support, obtaining engineering buy-in, overcoming resistance to change, and handling schedule and resource requirements. The third chapter provides support for the reader in handling the ongoing organizational support structure, including suggestions for governance, and continuing support from the management and the people involved in the project. Risks involved in new product development are enumerated, and suggestions provided for success metrics that can enable the team and management to assess progress and, ultimately, success in managing those risks.

The next three chapters delve further into the risks and opportunities involved in the project—topics that might be discussed just before fully launching the new product development effort. Chapter 4 elaborates on the DFSS process and discusses how both the software and hardware development efforts can be aligned. Chapter 5 delves into the business case risk in more detail, and provides a method for assessing the risk and opportunity in the context of a product roadmap that includes a portfolio of potential new products, and how the portfolio can be prioritized in the common situation of resource constraints. Chapter 6 discusses the project schedule with the associated, ever-present schedule risk, and provides some perspective, strategies, and approaches to handle schedule risk. These tools and methods include Monte Carlo simulation for the business case and the project schedule and theory of constraints project management/critical chain as a potential approach for handling schedule risks with respect to the project schedule.

The next several chapters are aligned with a flowchart that provides a step-by-step process for developing systems, software, hardware, or a combination of software and hardware. Chapters 7, 8, and 9 address the approach to gathering the VOC to understand what is important to customers, and the determination of requirements and selection of architecture based on customer and business expectations (VOC and VOB). This sequence of steps includes gathering, understanding, and prioritizing the VOC, making architecture decisions, and selecting measurable critical parameters requiring intense focus. Tools and methods include VOC gathering, KJ analysis, Kano analysis, QFD/House of Quality, concept generation methods, Pugh concept selection, and ATAM for software architecture decisions.

Chapters 10 through 12 discuss the "flow down" of the system-level requirements to requirements for hardware and software subsystems. The alignment of DFSS with Agile processes and software architecture selection are among the key topics involved in the software side, along with the engagement of rest engineering resources to ensure that the requirements can be measured, tested, and evaluated, as well as supply chain resources toward assuring supply chain readiness.

Chapters 13 through 15 discuss the concept of predictive engineering along with the optimization and "flow up" for meeting requirements allocated to both the software and hardware aspects. Methods including FMEA and fault tree analysis (FTA) help to anticipate and prevent problems. For continuous and ordinal requirements, a detailed selection process is provided for determining the transfer function for the critical parameters using a variety of methods relevant to both continuous and discrete variables. These methods include regression, logistic regression, DOE (design of experiments), RSM (response surface methodology), and robust design and stochastic optimization approaches to build high confidence that critical parameters will meet the customer's expectations. Chapter 14 also introduces an approach called Yield Surface Modeling that has had a remarkable success rate in terms of first-pass successes with high yields.

Chapters 16 through 19 correspond to the need to "trust but verify"; models and transfer functions are useful, but there is inherent risk in trusting that the model truly and completely represents reality. These chapters discuss verification and test, in which the capability and reliability of the software or hardware product is assessed on pilot and early production samples. Approaches include accelerated life testing (ALT) for hardware and fault injection testing for software reliability. The supply chain resources anticipate and take preventative action for potential supply chain issues, and verify that the supply chain, including vendors and internal manufacturing and testing facilities, is ready. Chapter 19 also introduces a novel statistical model for supply chains that can be used to attain goals for on-time delivery and quoted lead times with minimal strategic inventory levels.

The final chapter summarizes the topics and challenges discussed in the book, and provides a "look forward" toward future directions for new product development.

Supplementary Material Provided through the Web Site

There is a Web site associated with this book, available at http://www.sigmaexperts.com/dfss/. This Web site provides an interactive DFSS flowchart, aligned with the organization of the chapters in the book and with the software and hardware development process, which allow the reader to see a high-level overview of a topic, then click on a specific topic and "drill down" to a more detailed flowchart to aid with decisions on what approaches to consider, and to a summary of each approach.

The Web site also provides templates and Excel spreadsheets that will help the reader apply some of the approaches described in the book. There are examples on both the hardware and software side, including codes. Additionally, exercises are provided, aligned with the related chapters, to reinforce concepts and allow practice for the readers.

If the reader is interested in certification, there are Excel templates that can be used for project charters and for planning and later summarizing the project, and a PowerPoint template for presentations to summarize the project and its impact.

The Web site also provides materials, PowerPoint slides, and Acrobat files that will enable the reader to introduce topics to their organization and assist in selling concepts to management and the people involved in development.

10 Requirements Flow-Down

Position within DFSS Flow

Requirements flow-down is aligned with the early part of the Design phase of the RADIOV process, which corresponds to the transition from the Measure to Analyze phases of the DMADV process of DFSS or from the Concept to Design phases of the CDOV process. The DFSS flowchart, which can be downloaded from http://www.sigmaexperts.com/dfss provides a high-level overview (Figure 10.1) of the sequence of steps that can be drilled down to detailed flowcharts, and further drilled down to summaries for key tools and deliverables within each detailed flowchart. Figure 10.2 is the detailed flowchart aligned with this chapter, showing the steps and methods involved in flowing down system requirements.

For a system involving both hardware and software, the flow-down for system requirements will result in software and hardware requirements, evolving to subsystem requirements and to subassembly requirements and requirements for components (software components and hardware components). The sequence of steps in the flow-down process is iterative, in the sense that the anticipation of potential problems, measurement system analysis, and initial design capability analysis will be first performed at the system level, then at the subsystem/subassembly level, and then at the component level, as illustrated with the iterative nature of the flowcharts in Figures 10.2 and 10.3.

Figure 10.3 starts with a set of high-level system requirements. The process of "systems design" consists of turning these requirements into a specification of the system. First, a concept or architecture must be specified at the system level that identifies the subsystems, the interfaces between them, and any interfaces to the outside

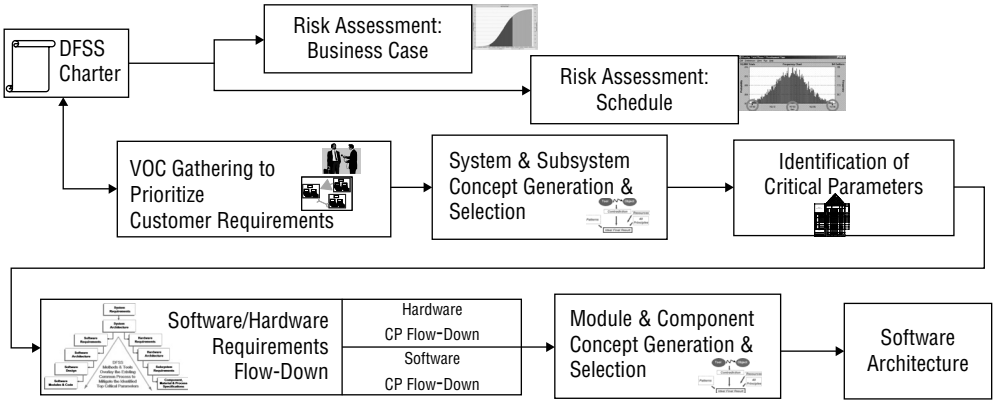


Figure 10.1 Flowchart overview highlighting step for flow-down of critical parameters

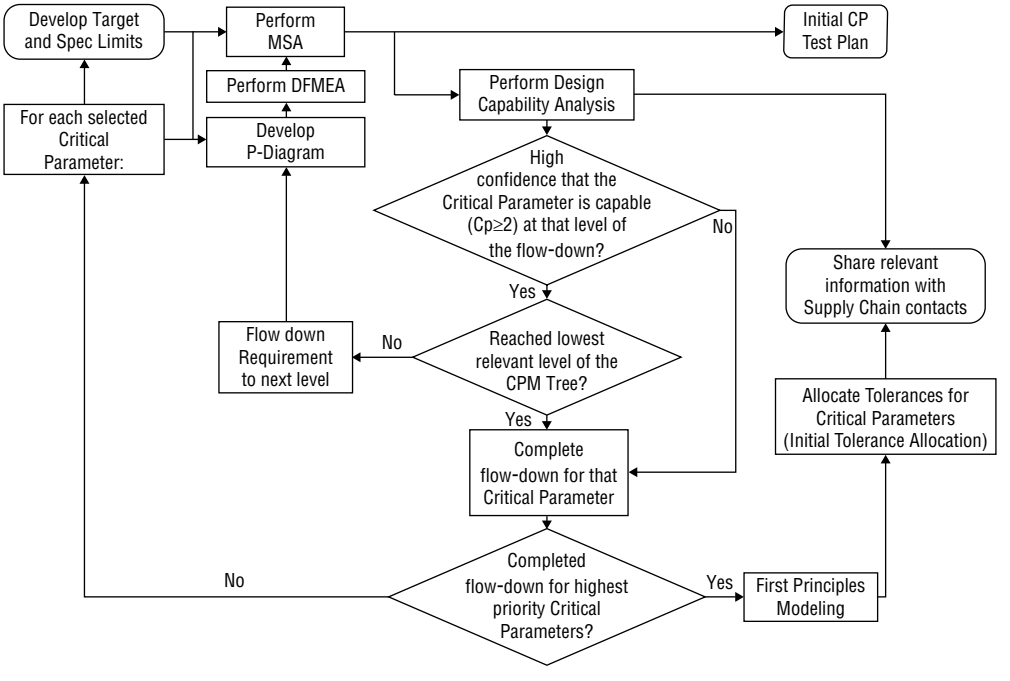


Figure 10.2 Flowchart, drilled down to detailed flowchart for flow-down of critical parameters

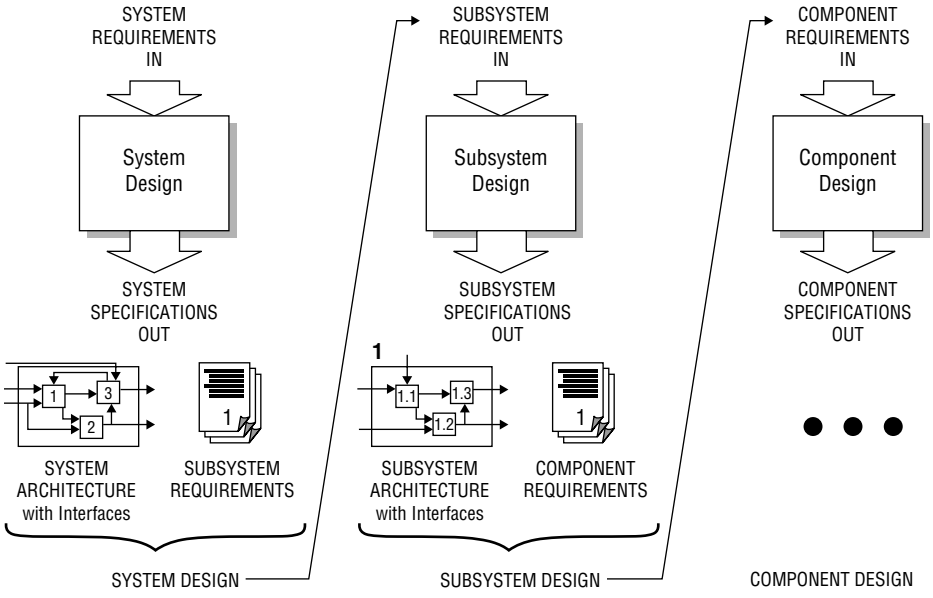


Figure 10.3 Evolution of system requirements

of the system—including the user interface and interfaces to or interactions with other systems. For electronics systems, interactions with other systems can be intentional, as in data communication linkages, or unintentional, such as with EMI (electromagnetic interference—unwanted disturbances caused by electromagnetic radiation emitted to or from another electronic system).

- For each subsystem, define the behavior and performance with subsystem requirements. Identify other subsystems and systems with which this system might interface or interact, and define the requirements for the interfaces.
- The team responsible for a subsystem is expected to not only deliver that subsystem so that it meets all of the requirements when isolated, but to meet all of the requirements when it is integrated with the other subsystems.
- The design for a subsystem requires considerations and decisions for how to meet all of the requirements jointly. This design should be documented in a subsystem specification that also contains the architecture for the subsystem, consisting of the components within the subsystem, the interfaces between these components, and the interfaces to other subsystems or other systems with which it can interact (see Chapter 12 for software architecture examples). Furthermore, each component has

requirements that define the behavior and performance required for the component to work with the other components and meet requirements jointly. This process continues down to low-level design.

FLOW-DOWN FOR HARDWARE AND SOFTWARE SYSTEMS

When developing a system comprised of both hardware and software, the flow-down of requirements to measurable technical requirements (subordinate *y*'s, *x*'s, and noises) might lead to three situations:

- Critical parameters that only involve hardware aspects
- Critical parameters that only involve software aspects
- Critical parameters that involve both hardware and software aspects.

For a hardware intensive system, or a system that involves little or no software, the critical parameters flow down to subsystems and components that are electrical or mechanical. The technical challenges for some products or some subsystems might fall entirely within one engineering discipline. For example, a team of mechanical engineers might flow down or decompose the critical parameters for a lawnmower to subsystem and component requirements. Other hardware intensive products might require a team composed of electrical and mechanical engineers, or the development organization might be structured so that these teams are separate, but a cross-functional team would handle the electrical-mechanical interactions and interfaces.

The flow-down for some requirements of a cell phone is particularly relevant for electrical engineers who are knowledgeable about radio frequency (RF) performance. Figure 10.4 shows part of the system-level House of Quality example discussed in Chapter 7. Two critical parameters, total radiated power (TRP) and turn-on time, are highlighted. In Figure 10.5, a second House of Quality focused on the RF sections of the cell phone indicates that TRP flows down to some measurable requirements for the antenna and for the transmitter. Figure 10.6 shows the flow-down, juxtaposed with some images of the physical layout within the cell phone.

Figure 10.7 shows the flow-down or decomposition of the critical parameter, cellular phone turn-on time, to hardware and software requirements. A team of system engineers, software engineers, and hardware engineers discussed this flow-down, and developed a simple mathematical model for the turn-on time, which showed that the delays in phone turn-on caused by the hardware requirements such as phase locked loop (PLL) lock time were negligible. The critical parameter for turn-on time then became a software development team focus.

Direction Of Goodness >		↑	↑	↑	↑	↑	↑	↓	↓	↑	↑	↓	↓	↓
Voice of Customer, VOC's	Importance	Display Size	Talk Time	Standby Time	Camera Resolution	Removable Memory Capacity	Application Processor MIPS	Bit Error Rate	Total Isotropic Sensitivity (T.I.S.)	Accurate Identification (Prob)	Baseband Processor (MIPS)	Phone Weight	Total Radiated Power (T.R.P.)	Turn-On Time
Fits lightly into a pocket or purse	6											9		
Can go 3 days without recharging	7		9	9										
Can email Quality photos easily & quickly	3	9	3	3	9	3	1	3	3		3			
Supports high speed internet connectivity	2	9					3	3	3		9			
Stores reasonable number of photos	1	1			3	9	3							
Few or no dropped calls	10		1	1			1	9	9		3		9	
Provides secure m-Commerce	1	3			3	1	3	1	1	9	3			
Turns on quickly	3													9
Scoring Totals		49	82	82	33	19	25	106	106	9	60	54	90	27
Target Nominal Values		60	10	50	5	2	16	0.000	−100	99.99%	100	4.5	30	10
Lower Limit		40	4	10	4	1	12	0.000	−120	99.9%	80	3.5	28.5	5
Upper Limit		100	40	1500	10	10	64	0.001	−95	100%	200	5.5	36	12
Units		mm	Hrs	Hrs	MPixels	MBytes	MIPS	%	dBm	%	MIPS	oz	dBm	Sec

Figure 10. 4 System-level House of Quality, focusing on two critical parameters, total radiated power from the transmitter and antenna, and turn-on time for the cell phone

Critical parameters that can involve both software and hardware aspects require that initial combined team approach. If software or hardware is totally dominant, then the effort can be handed off to the appropriate team as was the case for the turn-on time for the cellular phone. If neither software nor hardware dominates to such an extent, the effort on the critical parameter can either continue to be addressed by a team consisting of system engineers and software and hardware engineers, or the software aspects can be handed to the software team and the hardware aspects can be handed to the hardware team. In the latter instance, the interfaces and interactions between hardware and software risk "falling in the crack," so an additional effort is required to consider these interactions and to integrate the hardware and software aspects. In many cases, emulation can be used to evaluate the software aspects without the final version of the hardware but, rather, an existing hardware platform modified to behave like and substitute for the hardware.

The second House of Quality, as shown in Figure 10.5, is one of several methods to flow down requirements. Other methods can use a P-diagram, as discussed in the next section, or brainstorming session with a set of engineers, including system engineers, to

System Requirements	Importance Score	Antenna					Receiver							Transmitter					
		Antenna Gain	Antenna NF	Antenna Interface Attenuation	Antenna Cost	Antenna Weight	Receiver Gain	Receiver Cascaded Noise	Receiver Cascaded	Receiver Current Drain	PLL Lock Time	Receiver Cost	Receiver Weight	Transmitter Gain	Transmitter Output Power	Transmitter Power Added	Transmitter Current Drain	Transmitter Cost	Transmitter Weight
Display Size	4																		
Talk Time	8	**M**		**M**			**M**			H	L			H	H	H	H		
Standby Time	8	L	L							H	L			H	H	H	**M**		
Camera Resolution	2																		
Removable Memory Capacity	1																		
Application Processor MIPS	2																		
Bit Error Rate	10	**M**	**M**	**M**			H	H	**M**					**M**					
(T.I.S.)	10	H	H	H			H	H	L										
Accurate Identification (Prob)	0																		
Baseband Processor (MIPS)	5																		
Phone Weight	5					**M**							**M**						**M**
BOM Cost	0				**M**							**M**						**M**	
Total Radiated Power (T.R.P.)	8	H		**M**										H	H	**M**			
Turn-On Time	2							**M**			H								
Scoring Totals		224	128	168	0	15	204	186	40	144	34	0	15	246	216	168	96	0	15
Normalized Scores		9	5	7	0	1	8	8	2	6	1	0	1	10	9	7	4	0	1
Target Values																			
Units		dB	dB	dB	$	g	dB	dB	dBm	mA	sec	$	g	dB	dBm	%	mA	$	g

Figure 10.5 Second or subsystem-level House of Quality, focusing on the radio frequency (RF) subsystems including the antenna, receiver, and transmitter for a cell phone

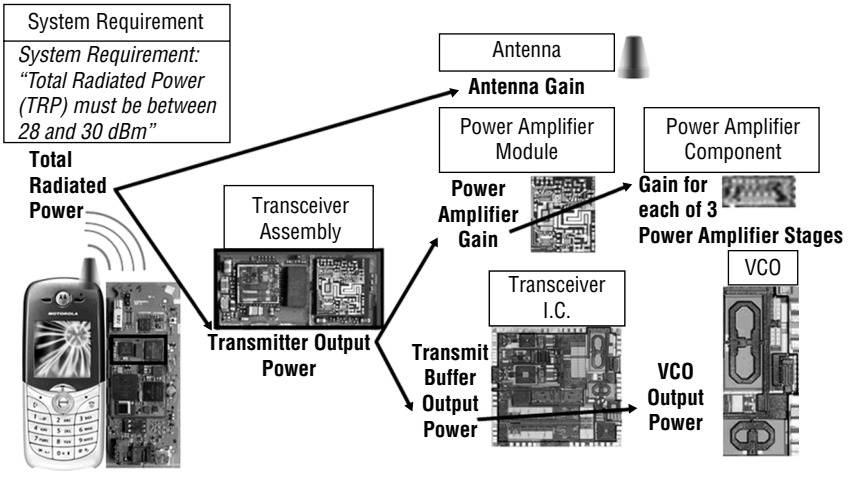


Figure 10.6 Flow-down of the total radiated power requirement for a cell phone to measurable requirements on the antenna and for the transmitter within the transceiver assembly

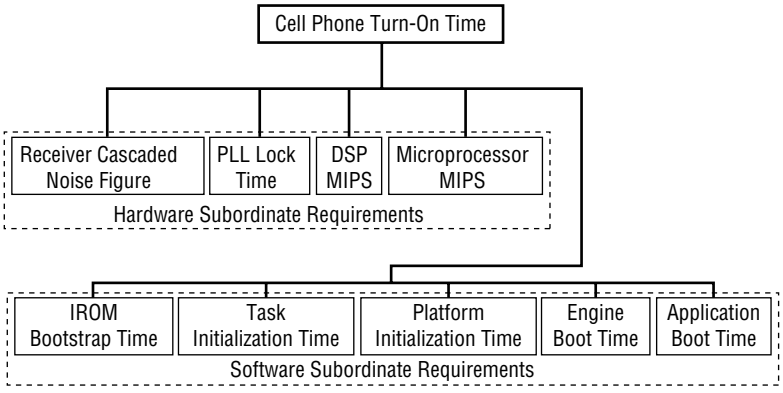


Figure 10.7 Flow-down of cell phone turn-on time to hardware and software requirements

identify indirect or intermediate requirements (subordinate *y*'s), control factors (*x*'s), and noises (*n*'s) that affect the critical parameter, as discussed in the flow-down section later in this chapter.

ANTICIPATION OF POTENTIAL PROBLEMS: P-DIAGRAMS AND DFMEA

System requirement flow-down also involves anticipation of potential problems. At the system level, a P-diagram and FMEA can be part of the concept generation and selection process for the system, as described in Chapter 8, and the system-level FMEA is included in the flowchart for identification of critical parameters, as described in Chapter 9.

As the flow-down proceeds iteratively, similar anticipation of potential problems should be subsequently applied for each critical parameter, or at the subsystem/subassembly level and the component level—and possibly at the manufacturing process level. Some perspectives should underline the importance of this anticipation as the flow-down proceeds: the system level flow-down will naturally involve a bird's-eye view of failure modes, and will involve a broader cross section of expertise for this purpose—but anticipation of failure modes and mechanisms at subsystem and module/component levels will involve a more focused set of experts to dissect the potential problems involved at that deeper, more detailed level. Essentially, at these subsequent iterations, the subsystem and component under consideration becomes "the system" for

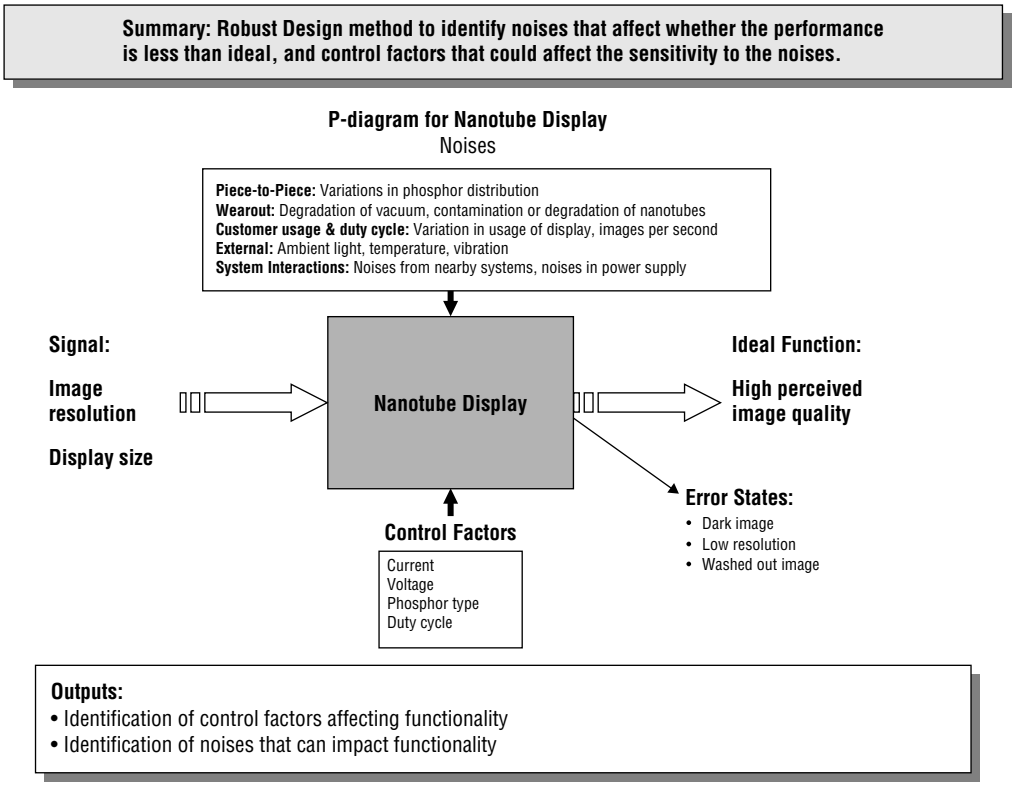


Figure 10.8 Tool summary for P-diagram

the team. It is worth noting that many of the subsystems for complex electronic products could literally ***be*** the "system" or product in other situations. For example, many cellular phones include digital cameras—but digital cameras are separate products or systems for camera manufacturers. Many cell phones ("music phones") incorporate music players, which also exist as separate products. In turn, many music players as products include flash drives, and some companies sell flash drives as products.

Either as an integrated subsystem, or as a separate product, anticipation of potential problems is a vital step toward prevention of problems. The P-diagram (Figure 10.8) and DFMEA (Figure 10.9) can be useful for the module and component-level concept generation and selection process described in Chapter 8, and also valuable in anticipating and preventing problems with aspects that were not selected as critical parameters (or flowed-down from critical parameters) but that could impact the success of the product if not adequately addressed.

Summary: A structured method for identifying and ranking the significance of various failure modes of the program and their effects on the product or customer

Failure Modes and Effects Analysis (FMEA):													Revised RPN			
System, Subsystem or Component	Parameter at System, Subsystem, Module, Assembly or Component Level	Potential Failure Mode / Effect	Potential Effects of Failure	SEVERITY	Potential Cause	OCCURRENCE	Current Controls	DETECTION	RISK PRIORITY NUMBER (RPN)	Action	Responsibility	Due Date	Severity	Occurrence	Detection	Risk Priority
				9		7		3	189				6	5	3	90
				8		7		4	224				6	6	2	72
				9		7		3	189				5	4	2	40
				9		7		3	189				5	3	2	30
				9		7		3	189				4	2	3	24
				9		7		3	189				4	4	1	16

1. List Functions and Interfaces or Subsystems.
2. List Potential Failure Modes.
3. List Potential Effects.
4. Assign Severity Rating.
5. List Potential causes.
6. Assign Occurrence Rating. (Probability rating for Software)
7. List current controls.
8. Assign detection rating.
9. Calculate Risk Priority Number.
10. Use RPNs to help decide on high priority failure modes.
11. Plan to reduce or eliminate the risk associated with high priority failure modes.
12. Re-compute RPN to reflect impact of action taken on failure mode.

Output:
- Ranked group of failure modes
- Impacts of failures
- Risk Priority Numbers (RPN) before and after corrective action
- Corrective actions, controls to remove or reduce the risk or impact of a failure mode

Figure 10.9 Tool summary for design failure modes and effects analysis (DFMEA)

A P-diagram (Figure 10.8) offers several benefits. It can help with the development of the DFMEA, in which the error states or deviations from the ideal function (at the lower right of P-diagrams) could suggest failure modes to be included in the DFMEA, and the noises (at the top of the P-diagrams) could suggest potential causes for the failure modes.

As will be seen later in this chapter, the team approach for identifying control and noise factors used in developing the P-diagram can be leveraged in flowing down requirements to the next level. The control factors portion of the P-diagram generally are the *x*'s in the flow-down, and the noises in the P-diagram obviously are the noises in

the flow-down. The missing pieces are the subordinate *y*'s—subrequirements that can be flowed down to other subordinate *y*'s, *x*'s, and *n*'s. If the critical parameter does not involve subordinate *y*'s, then the P-diagram can be used for the flow-down. However, many critical parameters cannot be directly flowed down to the final control factors with one P-diagram, and the P-diagram just provides a good start.

The P-diagram can also prove useful in generation and subsequent evaluation of alternative concepts for the subsystem, module, or component, particularly in terms of considering the noises that can affect performance when brainstorming potentially robust design approaches—the relative insensitivity of the alternative concepts to those noises can and should be considered in selecting a superior concept for the subsystem, module, or component.

The P-diagram can also prove valuable during transfer function determination (Chapter 13), in terms of initializing the identification of control and noise factors to use in an experimental design approach. The P-diagram will also prove valuable during optimization (Chapter 14), for evaluating and optimizing robustness against the noises. Some of the noises from the P-diagram can also be used as stress factors or for verification of reliability (Chapter 17).

FMEA (including system FMEA and design FMEA or DFMEA) has been discussed in Chapter 9, but it will be briefly reviewed here. The objective of DFMEA (summarized in Figure 10.9) is to consider the ways a product, subsystem, function, or interaction can fail, then analyze the risks, and take action where warranted. Typical applications include preventing defects, improving processes, identifying potential safety issues, and increasing customer satisfaction. It can be applied throughout the development life cycle. To be more effective, the DFMEA should relate to the nature of the development process itself. In either case, it considers overall architecture and functionality problems while at the same time addressing process problems. Therefore, DFMEA is an effective engineering tool for evaluating systems at a number of stages in the design process.

DFMEA evaluates risks posed by potential failure modes by considering the severity of the impact if the failure mode occurred, the probability that the failure mode could occur (based upon the probabilities for occurrences of potential causes of the failure mode), and the possibility that the problem would be detected in time. These three aspects of the risk are rated on a scale of 1 to 10, and then multiplied to provide RPN indices (on a scale of 1 to 1000) that can be treated as numerical assessments of risk. DFMEA and associated assessments are performed in a team setting, the atmosphere for which can become rather intense. It has been suggested that the DFMEA process be broken into two to three shorter sessions, during which the team is locked in a meeting room, and necessities (drink, raw meat . . .) are tossed over the wall.

There are systems that are heavily software oriented and that could benefit from a software DFMEA effort. The objective of a software DFMEA is to identify all failure

modes in a software artifact. Its purpose is to identify all catastrophic and critical failure probabilities so they can be minimized as early as possible. For example, a common problem in software involves memory leaks. A memory leak is an unintentional memory consumption by a computer program where the program fails to release memory when no longer needed. Memory is allocated to a program, and that program subsequently loses the ability to access it due to program logic flaws. A memory leak can diminish the performance of the computer by reducing the amount of available memory. Eventually, too much of the available memory may become allocated and all or part of the system or device stops working correctly, the application fails, or the system slows down unacceptably. For example, code that has a "***malloc***" (a subroutine for dynamic memory allocation) or a "***new function constructor***," which is evaluated each time it is encountered, can increase the risk of creating a memory leak. Memory leaks can corrupt and misalign pointers (which reference values stored elsewhere in memory), and may cause part or all of the system to go down; the system may have difficulty recovering, and in severe cases, key data may be lost.

DFMEA and P-diagrams can be used and reused through many of the subsequent steps of DFSS. This continuing value is realized because DFMEA and P-diagrams, in concert, help the team conceptualize and share an understanding of the risks by assessing the risks in terms of the severity or impact, the probability of occurrence, and the opportunities for errors. The team can also gain insight into noises as potential sources of variation, stresses, and failures.

Target and Spec Limits

Target values or specification limits for the critical parameters might have been developed as part of the QFD/first House of Quality effort, as discussed in Chapter 7. The specification limits are involved in the calculation of the P/T ratio in the measurement system analysis, the design capability analysis, and the tolerance allocation topics discussed in later sections of this chapter.

If the critical parameter is a lower-is-better type parameter, then it will generally just have one specification limit, the maximum. Examples of such one-sided parameters include leakage currents, defects or defect densities, costs, weight, delay times, and power consumption. The target in this situation could be half of the specification limit or maximum, or perhaps an achievable low value that would represent a value considered desirable for the customers.

Similarly, if the critical parameter is a higher-is-better type parameter, then it will have one specification limit corresponding to the minimum. Examples include battery life, drops-to-failure, mean-time-to-failure (MTTF), efficiency, and resolution. Some of these

examples are bounded on both sides by the nature of the metric or by physics; for example, percent efficiency is bounded by 0 and 100 percent, even though it is considered a higher-is-better type parameter. The target in this situation could be twice the lower specification limit, or an achievable high value that would be considered desirable for the customers.

If the critical parameter is a target-is-best type parameter, then it will have both an upper and a lower specification limit. Examples could include total radiated power (TRP) for a transmitted signal and some timing requirements in a clocked system constrained by issues such as race conditions. Generally, for two-sided limits, the target will be midway between the upper and lower specification limits; however, there will be exceptions to this, such as situations where the critical parameter is believed to follow a lognormal distribution, in which case the target might be the geometric average of the upper and lower specification limits (that is, the square root of the product of the upper and lower specification limits). Alternatively, the target is an achievable value that would be considered desirable by most of the customers; ideally, if the manufacturer could produce all parts with exactly that value, the customers should be satisfied (if not downright ecstatic).

Companies are rife with examples of problems with measurement systems analysis (MSA), capability indices, SPC, and customer issues that trace back to specification limits set arbitrarily, such as to some target ± 10 percent. The specification limits should be based on what is needed to meet the customers' expectations—and, subject to that consideration, the spec limits should be as wide apart or as generous as reasonable for the design team. This enables the design team to have the best chance of success in meeting the specifications with high confidence, and creates a high likelihood that the customers will be satisfied by the result of the design teams' innovation, optimization, and robust design of the product.

In some instances, appropriate specification limits may be hard to pin down. One possible cause for this fuzziness might be that different customers or sets of customers may have different expectations. There are at least three alternative approaches that can be used to deal with this issue: the best and widest-spaced compromise can be selected to satisfy the largest groups of key customers, the product can become multiple products each tuned to the expectations of different customers, or the characteristic can be designed to be tunable, programmable, or selectable by the customers.

MEASUREMENT SYSTEM ANALYSIS

Once the critical parameters have been selected, and specification limits have been set, it seems reasonable that the next steps might be to set things up so that progress towards achieving expectations can be monitored. As discussed in Chapters 7 and 9, the critical parameters have been defined in measurable terms. The next logical step is to set up the

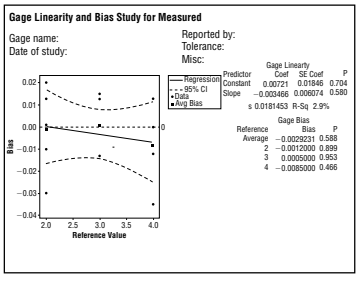
measurement systems and determine whether each is capable of measuring appropriate critical parameters.

Figure 10.10 summarizes the purpose, results, and outputs from measurement system analysis, focusing on MSA for critical parameters that are continuous rather than discrete. There are several indices used to determine if the measurement system is adequate for the purposes of optimization and validation of the critical parameters, including assessments of stability, linearity, accuracy, and measurement error. MSA is discussed further in Chapter 16.

The assessment and estimate of measurement error is a key, recurring topic in DFSS, and this is an appropriate point to begin that discussion. The measurement error is one of several "noises" that can be flowed down, as discussed later in this chapter, and that will be encountered along the way as the design team uses approaches such as design of experiments (DOE) and response surface methodology (RSM).

This aspect of the flow-down process is illustrated in Figure 10.11, which starts with the concept of squared deviation from the target. If the target is the desired value, as discussed in the previous section, then one can define a statistical index, the second moment about the target, which can represent the degree of customer satisfaction.

Summary: Statistical analysis of the variation caused by a measurement system and documenting Precision, Accuracy, and Capability of measurement systems.



Gage R&R for Diameter

Source	StdDev (SD)	Study Var (6 * SD)	%Study Var (%SV)	%Tolerance (SV/Toler)
Total Gage R&R	0.115544	0.69326	67.32	34.66
Repeatability	0.115544	0.69326	67.32	34.66
Reproducibility	0.000000	0.00000	0.00	0.00
Operator	0.000000	0.00000	0.00	0.00
Part-To-Part	0.126931	0.76158	73.95	38.08
Total Variation	0.171644	1.02986	100.00	51.49

Number of Distinct Categories = 1

% GR&R = 67.32 %P/T = 34.66

Outputs:

- Assessment of Stability
- Estimate of Accuracy (Bias)
- Estimate of Linearity
- Estimate of Measurement error, Std dev of Repeatability (within same conditions) and Reproducibility (operator-to-operator)
- Assessments of Precision: Precision-to-Tolerance (P/T) Ratio and Gage R&R

Figure 10.10 Summary for measurement system analysis

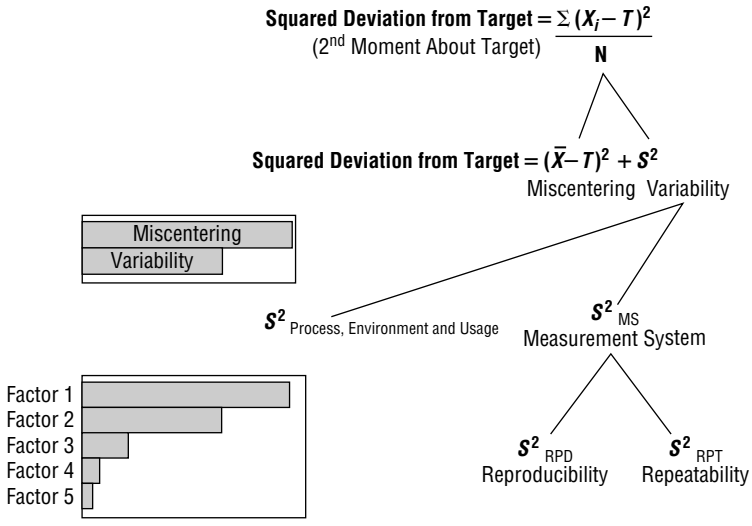


Figure 10.11 Partitioning of squared deviation from the target, including variance associated with the measurement system

The ideal case, in which every product is exactly on target with no variation, would have a value of zero for this statistical index. As further illustrated in Figure 10.11, this squared deviation from (or second moment about) the target corresponds to the Taguchi Loss Function for a target-is-best situation. A useful aspect of this equivalence is that the deviation from the ideal situation can be partitioned into two parts: the degree to which the deviation is a result of the average being off-target, and the variance about the mean. This variance can be further partitioned into variance as a result of the measurement system (discussed here) and variance as a result of manufacturing variation and variations in usage and environment (including system interactions).

MSA for continuous parameters provides an estimate for the variance caused by the measurement system, and compares it to the tolerance in terms of the precision to tolerance ratio (P/T ratio), and to the total observed variance in terms of the GR&R ratio (gauge repeatability and reproducibility). The P/T ratio is defined as six times the standard deviation of the measurement system divided by the difference between the upper and lower specification limits. The GR&R ratio is defined as the standard deviation of the measurement system divided by the total observed standard deviation, combining sources of variation including measurement error, variation from manufacturing, variation from how the customers use it, variations from the environments where the product will be used, and variations in how the interactions among the subsystems and the product with other systems affect the parameter.

If the measurement variance consumes too much of the tolerance window, or obscures the ability to assess the other sources of variation, then the measurement system is not acceptable. For many situations, the rule of thumb is that both the P/T ratio and the GR&R ratio should be less than 30 percent; for other situations, a rule that both should be less than 10 percent is imposed. Acceptable values for the P/T ratio derive from statistical analyses that indicate that a P/T ratio more than 30 percent corresponds to a very high risk of incorrectly passing bad parts or incorrectly rejecting good parts.

The measurement system for critical parameters at the system or product level will generally link to the test and evaluation plan for the product, as illustrated by the arrow to the deliverable initial critical parameter (CP) test plan in Figure 10.2. This deliverable is a starting point for the verification of capability discussed in Chapter 16, and summarized in Figure 10.12. Clearly, the preparation of the measurement systems to be used

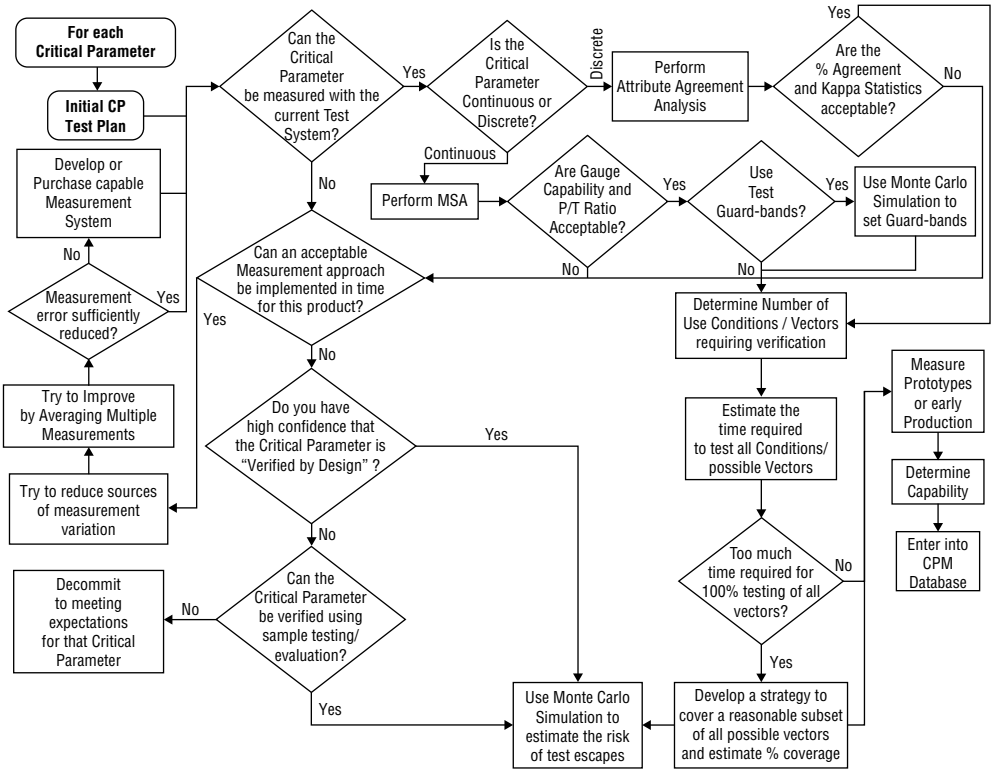


Figure 10.12 DFSS flowchart, drilled down to detailed flowchart that includes actions for improving the measurement system if MSA results are unacceptable

for verification do not need to wait, and should not wait, but should be initiated with the initial measurement systems analysis effort.

If the GR&R or the P/T ratio, or both, fail to meet acceptable guidelines, then there are a series of actions for improving the measurement system that are summarized in Figure 10.12 and discussed in Chapter 16.

CAPABILITY ANALYSIS

The next step shown in Figure 10.2 involves preliminary assessment of design capability. The design capability is a predicted capability, as opposed to the measured capability assessed on existing products or processes in the DMAIC process improvement flow. The preliminary assessment is performed in the Design phase of RADIOV (corresponding to the Measure phase of DMADV); if it is inadequate, then later steps (largely in the Optimize phase of RADIOV or Analyze through Design phases of DMADV) will improve the capability and a new assessment of the design capability will presumably be reflected in an improved value for the design capability indices. Later, in the Verify phase, the actual capability will be assessed on prototypes or early production samples, as discussed in Chapter 16.

As a predicted capability, the design capability might be assessed using predictive methods such as Monte Carlo simulation or a method referred to as the propagation of errors or system moments method in some situations and the root sum of squares method in other situations. These predictive engineering methods are discussed in Chapter 14.

There are two key indices used to assess design capability: the *Cp* (also known as *Pp*) and the *Cpk* (also known as *Ppk*). Equations for these two indices are given here:

$$Cp = \frac{USL - LSL}{6s} \tag{10.1}$$

$$Cpk = \min\left[\frac{USL - xbar}{3s}, \frac{xbar - LSL}{3s}\right] \tag{10.2}$$

Six Sigma performance is defined as having a *Cp* greater than or equal to 2 and a *Cpk* greater than or equal to 1.5. It is possible that the initial design capability assessment will forecast *Cp* and *Cpk* values that meet Six Sigma performance expectations at the get-go. If the team has confidence in this initial estimate, the design team can breathe a sigh of relief, celebrate, party, and paint the town red as appropriate to their personalities and local laws and customs. In addition to this emotional reaction, the design team need not expend any further effort on this critical parameter unless something changes

that would jeopardize this pleasant state of affairs. Consequently, the detailed flowchart in Figure 10.2 shows that the flow-down for that critical parameter can be considered complete, and the design can move on to the efforts for the next critical parameter.

In those cases in which the initial assessment of the design capability do not provide sufficient confidence that the initial design is capable, the next step would entail flow-down or decomposition, as discussed in the next section.

Flow-Down or Decomposition

If the initial design capability analysis does not provide high confidence that the critical parameter will reside comfortably and consistently within the specification window, robust against noises ranging from manufacturing variation through variations in use conditions, environments, system interactions and measurement error, then the team will need to engage in robust design and optimization efforts that will generally be performed at the subsystem, module, subassembly and/or component levels. Consequently, the next steps involve identifying the parameters at these levels that are affecting the performance of the critical parameter at the system level. This is referred to as the critical parameter flow-down process.

A valuable tool for critical parameter management in general, and for this critical parameter flow-down and the later process for critical parameter flow-up (Chapter 14) is called Cognition Cockpit (http://www.cognition.us). This software tools provides an easy-to-use, Web-based interface that handles virtually all aspects of critical parameter management and provides interfaces to other software commonly used in DFSS and in product development.

The critical parameter flow-down is a team activity involving the appropriate expertise to identify *x*'s, *n*'s, and subordinate *y*'s that affect the performance of the system level critical parameter. The second House of Quality can help with this flow-down. The approach used previously in the system-level or first House of Quality, described in Chapter 7, would be used again at the subsystem level or the next level down in the product hierarchy, but with the measurable system-level parameters along the left side and subordinate measurable technical parameters for the subsystem described across the top, as in Figure 10.5. Although this approach tends to provide a useful set of subordinate *y*'s for each subsystem, the control factors (*x*'s) and noises (*n*'s) are a bit more difficult to obtain from this method.

The term "*x*'s" refers to factors that are under the design engineers' control: design choices, component choices, or settings of continuous variables, like the choice of a resistor or capacitor value or for a voltage-controlled oscillator or the setting on a voltage supply.

The term "*n*'s" refers to noises: factors that will not be under the design engineers' control when the product is operating out in the field among customers. Noises like

environmental temperature might be controllable in the lab environment, which will be useful for evaluation purposes, but cannot be controlled once it leaves the controlled environment—a customer may use the product during a summer in Phoenix, Arizona, or Riyadh, Saudi Arabia, and the same or a different customer may use the product during winter in Alaska or Sweden.

The term "subordinate *y*'s" refers to measurable parameters at a lower level in the flow-down that affect the system-level performance for the critical parameter and are in turn affected by other factors and parameters at an even lower level in the flow-down. A mechanical example of this might be the water resistance of the system being flowed down to subordinate *y*'s representing the water resistance of various inserts, holes, and user interfaces that cannot be affected directly but can be affected indirectly through the choices of O-rings and dimensions that can ensure acceptable water resistances for those subordinate *y*'s. An electronic example might be the total isotropic sensitivity, corresponding to the weakest signal strength that the system can dependably handle, which can be flowed down to subordinate *y*'s representing the antenna gain and the LNA (low noise amplifier) gain at the component level, which cannot be directly affected but can in turn be affected by decisions about the design of the antenna and selection of the LNA part or components in the LNA.

The flow-down effort can be facilitated by the use of the P-diagram (Figure 10.3) discussed earlier in this chapter, which could already have identified the noises and may simply require differentiation between the subordinate *y*'s and the *x*'s. Alternatively, the team can use a second House of Quality approach or participate in a meeting to brainstorm the factors that affect the critical parameter and subsequently differentiate the factors as noises, *x*'s, or subordinate *y*'s, with an additional step to further explore the subordinate *y*'s to complete the flow-down to *x*'s and *n*'s.

The process described here has proven very efficient at quickly generating a more thorough first-pass flow-down, which can subsequently be refined and expanded or "fleshed out." It also can be used to quickly generate P-diagrams and subsystem or component Houses of Quality or entered into the Cognition Cockpit database.

Procedure for Critical Parameter Flow-Down or Decomposition

1. Ask the critical parameter owner to describe the critical parameter, how it's measured, current estimates about its most likely value and possible distribution, and any progress that's already been made towards developing confidence that the critical parameter will be capable.
2. Discuss with the team:
   Is the critical parameter clearly measurable as-is? How would it be measured?
   If the measurement approach is clearly defined, would meeting that measurable

requirement fulfill customer and business expectations? If not, develop an operational definition, a measurable definition for the critical parameter.

3. Brainstorm subrequirements with the team, first pass. The template shown in Figure 10.13, available for download at http://www.sigmaexperts.com/dfss/, can help with this process.
4. Classify the subrequirements into subordinate *y*'s, *x*'s, and noises.

   Subordinate *y*'s are measurable, but not directly controllable (i.e., there is not a "knob" to change the value of the subordinate *y* to a selected value).

   Control factors or *x*'s are directly controllable and affect the value of the critical parameter or a subordinate *y* to the critical parameter.

   Noises or *n*'s are factors that affect the critical parameter or a subordinate *y*, but that the team does not control in normal usage (although the team might be able to control a noise like temperature in a lab).

Subordinate *y*'s (Subsystem, Module, or Component)—Measurable Requirements:	Priority	Units	*y*, *x* or *N*	Continuous or Ordinal Subrequirements:	Binary or Obligatory Subrequirements:	Is this Necessary to Fulfill the Requirement?	Is this Set Sufficient?
						Yes	**Yes**
						Yes	
						Yes	
						Yes	**No**
						No	
						Yes	
						Yes	**Yes**
						No	
						No	

Figure 10.13 Template for critical parameter flow-down process and for associated P-Diagrams; this template can be downloaded at http://www.sigmaexperts.com/dfss/chapter10flow-down

5. Classify the subordinate *y*'s into continuous requirements, ordinal requirements, and binary discrete or obligatory requirements (pass/fail or meets/doesn't meet requirements).

   Continuous requirements have a full range of possible values—like a voltage measurement; ordinal requirements have integer values, where higher or lower is better—like a score of 1 to 7 on a Likert survey form, or the number of drops to failure, or the number of clicks to get to a certain screen.

   Binary requirements are either acceptable or unacceptable—like whether an Excel-compatible table of data is output from the software or not.
6. For each subordinate *y*, ask the team—is each subrequirement necessary?

   If this set of subrequirements was satisfactorily met, would that provide sufficient confidence that the product will meet expectations for the critical parameter? If not, brainstorm what additional subordinate *y*'s are needed to have sufficient confidence that the customers will be satisfied that this critical parameter has been fulfilled.
7. For each necessary subordinate *y* that is continuous or ordinal, the team should discuss their confidence. If the team is highly confident that a subordinate *y* will be satisfactorily achieved, then it need not be flowed down further, but otherwise the team would brainstorm other lower level subordinate *y*'s, control factor (*x*'s) and noises (*n*'s) that affect that subordinate *y*. Continue until the lowest level of the flow-down consists of only *x*'s, *n*'s, and subordinate *y*'s that the team is highly confident will be satisfactorily achieved.
8. For each necessary subordinate that is binary or obligatory, ask the team: What are the goals? Should we consider a Pr (success) metric (as discussed in Chapter 14)? Would fault tree analysis be helpful for this obligatory requirement? (Fault tree analysis is discussed in Chapter 9.)
9. Put the results of the flow-down of the critical parameter into a diagram, with the critical parameter or *Y* placed at the top or left side of the diagram, and the subordinate *y*'s, *x*'s, and *n*'s linked by lines or arrows. If appropriate, capture this flow-down in a database such as the critical parameter management database associated with Cognition Cockpit. If appropriate, capture part or all of the flow-down as a P-diagram.
10. Ask the team or assign team members to obtain goals/preliminary spec limit(s) for the continuous and ordinal requirements.

FLOW-DOWN EXAMPLES

The flow-down described in this chapter can be applied to a variety of parameters, including mechanical, electrical, and software parameters. In Figures 10.14, 10.15, and 10.16, qualitative flow-down will be applied to three critical parameters for a set of

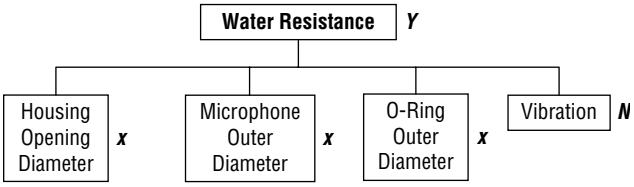


Figure 10.14 Critical parameter flow-down of water resistance for "Simon" communication device

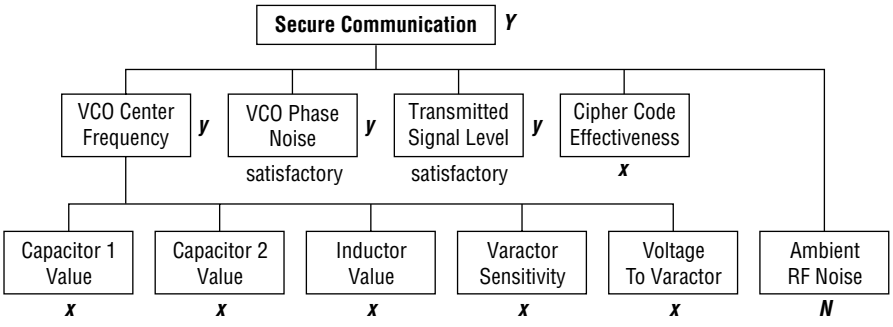


Figure 10.15 Critical parameter flow-down of secure communication for "Simon" communication device

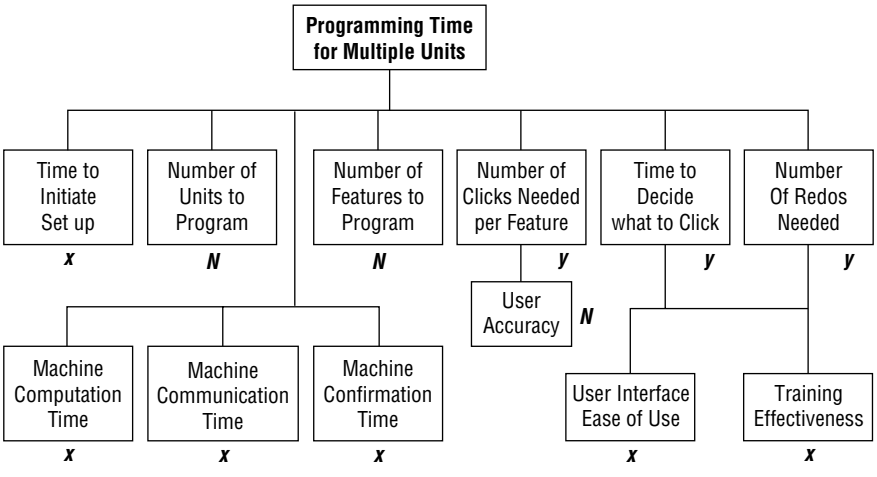


Figure 10.16 Critical parameter flow-down of programming time for multiple units

communication devices for secret agents; the device is code-named "Simon." These critical parameters are water resistance, secure communication, and programming time for multiple units. By sheer coincidence, the first critical parameter is primarily a mechanical engineering example, the second is a combined software and electrical engineering example, and the third is primarily a software example.

Water resistance is largely dependent on the materials used in the housing (outer shell) of the communication device and the effectiveness of the seals involved in the opening in the housing to accommodate a microphone. The mechanical engineering design team is confident that the housing itself is impervious to water intrusion, and the primary risk is seal for the microphone. The team has identified the x's as the housing opening diameter, the microphone outer diameter, and the outer diameter of an O-ring that must not exhibit excessive compression. The design team is also concerned with vibration as a noise. The ultimate customers (spies, secret agents, and informers) are prone to considerable vibration in the usage environment, as the team ascertained through exhaustive research (watching James Bond movies; popcorn optional). This example is similar to an actual DFSS project, which is shared as an example for optimization and flow-up in Chapter 14.

The design team flowed-down secure communication (measured by a secure communication effectiveness metric) to a software parameter (cipher code effectiveness) and several subordinate *y*'s, including the center frequency of the voltage-controlled oscillator (VCO). This subordinate *y* was flowed down to parameters associated with the two capacitors, an inductor, a varactor, and the voltage applied to the varactor, as shown in Figure 10.15.

The team also flowed down the requirement for the time required to program a set of communication devices, as shown in Figure 10.16.

INITIAL TOLERANCE ALLOCATION

After the flow-down has been completed in a qualitative respect and specification limits and a target value for a system-level critical parameter have been set, the next step is to allocate tolerances from the critical parameter to the subordinate *y*'s and *x*'s involved in the flow-down.

Initial tolerance allocation is the quantitative part of the critical parameter flow-down. Tolerances may be available from suppliers for some of the subordinate y's and *x*'s. For others, the initial tolerance may be the start of communication with the suppliers and assembly and manufacturing areas involved in the supply chain. If the supplier already has a proposed tolerance, it could be helpful for the engineering team to compare the tolerances proposed by the suppliers to a baseline to ascertain whether the suppliers' tolerances align with reasonable or expected tolerances.

	Units	LSL	Target	USL	kmin	kmax	
Frequency	**MHz**	**2.4**	**2.5**	**2.7**	**0.07**	**0.13**	
Subordinate *y*'s	**Units**	**Slope**	**Target**		**Allocated LSL**	**Target**	**Allocated USL**
Capacitor 1	pF	–0.24793	**2**		**1.9**	**2.0**	**2.3**
Capacitor 2	pF	–0.24793	**1**		**0.9**	**1.0**	**1.1**
Varactor Sensitivity	pF/V	–0.48868	**1**		**0.9**	**1.0**	**1.1**
Inductor	nH	–0.15582	**8**		**7.5**	**8.0**	**9.0**
Voltage	V	–0.24793	**2**		**1.9**	**2.0**	**2.3**

Figure 10.17 Excel worksheet for quantitative flow-down of allocated tolerances

The approach described in this section can be used for a variety of situations:

- **Schedule allocation**: Starting with best case/most likely/worst case durations for developing features, and allocating these to subtasks required to develop the feature.
- **Timing/delay allocation**: Starting with a range or mean and standard deviation for overall timing for a feature or function, and allocating the total timing or delay to the individual tasks.
- **Mechanical tolerance allocation**: Allocating tolerances that are additive, like tolerances for components in a gap analysis.
- **Electrical tolerance allocation**: Allocating tolerances for a function to its subfunctions when the transfer function is not necessarily additive and some of the subordinate *y*'s or *x*'s might be in different units than the *Y* (for example, the *Y* may be frequency in MHz and the *x*'s might be capacitance in pF and inductance in nH).

A step-by-step approach for allocating tolerances to the subordinate *y*'s and *x*'s is provided here. The Excel template shown in Figure 10.17 can be downloaded to assist with these calculations. The subordinate *y*'s and *x*'s are both referred to as subordinate *y*'s and treated the same in this approach. If the transfer function is a simple additive or sum of terms function, then the slopes will be set to unity. If the transfer function is a simple multiplicative or product of terms function, then a logarithmic transform could allow the same approach to be used, with the slopes similarly set to unity.

1. Determine Tolerance for *Y*: Target, USL and LSL.
2. Determine Target or Most Likely Values for subordinate *y*'s: $Y_{T,i}$ and slopes: $b_i = dY/dy_i$.

3. Estimate the constant percent tolerance for each subordinate y_i:

$$k_{\max} = \sqrt{\frac{(USL - T)^2}{\sum (b_i y_{T,i})^2}}$$

4. For each subordinate y, set $\mathrm{USL}(y_i) = y_{T,i}\,(1 + k_{\max})$.
5. If the Tolerance for Y is symmetrical, set $\mathrm{LSL}(y_i) = y_{T,i}\,(1 - k_{\max})$.
6. If the Tolerance for Y is not symmetrical:
   a. Determine the constant percent tolerance for each subordinate y to its lower limit:

$$k_{\min} = \sqrt{\frac{(T - LSL)^2}{\sum (b_i y_{T,i})^2}}$$

   b. Set $\mathrm{LSL}(y_i) = y_{T,i}\,(1 - k_{\min})$.

As illustrated in Figure 10.2, these allocated tolerances should be shared with suppliers and manufacturing and assembly engineers, or with supply chain experts who can work with suppliers and manufacturers.

The quantitative aspect of the flow-down described in the previous section will be applied through an Excel worksheet set up as a template, as shown in Figure 10.17. Figure 10.15 includes a subordinate y called "center frequency." The quantitative flow-down for the subordinate y of center frequency for the VCO is illustrated in Figure 10.17, using the template that can be downloaded from http://www.sigmaexperts.com/dfss/chapter10allocation. The transfer function for the center frequency is not additive; it is a constant divided by the square root of the product of the inductor value and the sum of the capacitances for the two capacitors and the varactor. This transfer function was evaluated to obtain slopes for frequency versus each factor for use with the spreadsheet template.

SUMMARY

The system requirements flow-down is the beginning of the Design phase of the RADIOV DFSS process, initiating activities for predictive engineering: to design and optimize the flowed-down parameters at the subsystem, subassembly, module, and component level in order to provide high confidence that expectations for the system-level critical parameters will be successfully met. The systems (software and hardware) requirements flow-down also initiates focused activities in the systems/field test and supply chain to help develop confidence in meeting the flowed-down requirements.

Index

Numbers

100-point rating method, for customer requirements, 125

A

Abstraction design heuristic, for software architecture, 228
Accelerated life testing, 328–330
Acceptability criteria, for success metric, 57
Acceptance testing, 359
Accuracy, MSA and, 307
Actors, in use case modeling, 295
Ad-hoc testing, 359
Affinity diagramming
  interviewing and, 114
  KJ analysis applying, 117
Affordances, in software mistake proofing, 300
Agile Alliance, 211–212
Agile development
  applying critical parameters to development of feature sets, 213–214
  data collection plan for Motorola ViewHome project, 219–220
  DFSS tools used with, 215–217
  handling changes in requirements, 389
  iterative development and, 212–213
  manifesto for, 212
  measuring agile designs, 218–219
  noise factors and, 217–218
  overview of, 211–212
  requirements gathering (VOC and VOB), 214
  schedule risks and, 106
  SDLC (software development lifecycle) and, 212
  summary, 221
  verification process, 218
Algebra, for deriving equations, 246
All-pairs or pairwise testing, 356–357
Alpha testing, 359
Altshuller, Dr. Genrich, 141
Alvarez, Antonio, 2–3
Analysis aspect, of risk management, 60
Analyzing concerns and issues, 34–39
  job titles and responsibilities and, 38–39
  overview of, 34
  resource requirements and, 36–38
  time required for DFSS, 35–36
  waiting out corporate initiatives, 36
  when product development process is already in place, 34–35

Anderson-Darling test, 315
Application framework (layer 3), software architecture design, 224
Application support services (layer 2), software architecture design, 224
Applications (layer 4), software architecture design, 224
Architecture
  alternate architecture generation for hardware and software, 143–146
  approaches to selecting, 49
  risks of changing legacy architecture, 230
  software. *See* Software architecture
Architecture phase, RADIOV
  in CDMA cellular phone example, 17–18
  overview of, 75, 77–78
  tools for, 77
Assembly, DFMA (Design for manufacturability and assembly), 366–369
Availability
  critical parameters and, 172–174
  flow-down, 321–322
  measuring, 320
  modeling case study, 342–346
  software architecture design tactics for, 229
  verifying. *See* Verification of reliability and availability
Axiomatic design, 145

B

Barriers, to acceptance of DFSS
  analyzing potential, 34
  existing processes as, 34–35
  job titles and responsibilities as, 38–39
  removing, 42–43
  resource requirement as, 36–38
  schedule risk as, 95
  time requirements as, 35–36
Baselines, success metrics and, 41–42
Bathtub curve
  early life failures/infant mortality and, 326
  reliability and availability and, 322–325
  useful life/constant failure rates and, 326–327
  wear out mechanisms and, 327
Bayes theorem, in WeiBayes approach to failures, 330–331
Behavioral design patterns, 354–355
Benchmarking, applying Six Sigma to process goals, 5
Beta testing, 359
Bill of materials (BOM), 155–156
Binary logistic regression, for defect reduction, 389
Black belts
  certification of, 62–63
  risk assessment role of, 352
Black box tests
  definition of, 359
  software testing, 347
  system testing, 348
BOM (bill of materials), 155–156
Box-Cox transformation, 315
Brainstorming
  for concept generation, 140–141
  feasibility screening and, 148
  as method to flow-down requirements, 191
  for VOC-responsive features, 214
Branches/conditions coverage, in end-to-end testing, 349
Buffer overflows, preventing in software mistake proofing, 303
Buffers, in critical chain project management
  personal, 104–105
  project, 105
"Burning platform"
  articulating, 33
  as means of overcoming resistance to change, 30–32
  supporting DFSS deployment, 29–30
Business case risks
  adjustments based on commitments and strategic direction, 92–94
  analyzing projects already underway, 93–94

goals, constraints, considerations, and distractions, 91–92
metrics for, 59, 85–89
overview of, 83–84
portfolio decision making as optimization process, 84–85
resource constraints and, 89–91
summarizing address to, 94
Buy-in, in concept selection, 149

C

Calculus, for deriving equations, 246
Calibration standard, MSA and, 309
Capability
design capability. *See* Design capability
predicting software release capability, 304–305
process capability. *See* Process capability
Causes, of failure
controls for, 162–163
listing, 160–161
Pareto analysis of cause categories, 171
CBAM (cost-benefit analysis method), 232–234
CCRs (critical customer requirements), 119
CCD (central composite design), 256–257
CDMA base station example, 14–26
Architecture phase, 17–18
Integration and Optimization phases, 18–21
IP-BSC (IP-based base station controller) and, 14–16
Requirements phase, 15, 17
Verification phase, 21–24
CDOV (Concept, Design, Optimize, Verify)
critical parameter flow-up and, 263–265
DFSS process nomenclature and, 69–70
DFSS steps, tools, and methods associated with, 12–13, 71–72
DOE (design of experiments) and, 251
identification of critical parameters, 153
as phase of DFSS project, 9
RADIOV phases corresponding to, 15
requirements flow-down and, 187
schedule risks and, 95
software verification testing and, 350
verification of design capability and, 307
verification of reliability and, 319
verification of supply chain readiness and, 364
Cellular phones
as example of new product development, 385
Center for Quality Management, 109
Central composite design (CDD), 256–257
Certification, 62–64
Champions. *See also* Leadership
certification of, 64
obtaining for DFSS projects, 46
removing roadblocks and impediments, 42
supportive project reviews and, 54–55
Change agents, certification of, 62–63
Change management
DFSS deployment and, 44
handling changes in requirements, 389
overcoming resistance to change, 30–32
Coalition, role of guiding coalition, 32
Code
end-to-end testing, 349
libraries, 224
unit testing, 347
Cognition Cockpit, 203
Combinatorial design method, 356–358
Combinatorial optimization algorithms, 356–357
Commitments, adjusting decisions based on, 92–94
Communication
risk management and, 60
of vision, 40–41
Comparison tests, 359
Compatibility testing, 359
Competitive analysis, building House of Quality and, 129
Component level, concept generation at, 139
Comprehensiveness criteria, for success metrics, 57
Compression design heuristic, for software architecture, 228

Concept, Design, Optimize, Verify. *See* CDOV (Concept, Design, Optimize, Verify)
Concept engineering, 109
Concept generation/selection
  alternate architecture generation for hardware and software, 143–146
  approaches to, 137–139
  brainstorming and mind-mapping, 140–141
  concept selection process, 49, 149–151
  consideration of existing solutions, 147–148
  developing feasible concepts to consistent levels, 148–149
  feasibility screening, 148
  flowchart for, 138
  Kansei engineering approach, 152
  position within DFSS flow, 137
  robust design concepts, 146–147
  summary, 152
  TRIZ, 141–143
Concerns, analyzing. *See* Analyzing concerns and issues
Conflict resolution, 55
Consistency, concept generation and, 148–149
Consolidating gains, in DFSS deployment, 44–45
Constant failure rates, reliability and availability and, 325–326
Constraints. *See also* critical chain (theory of constraints) project management
  critical parameters and, 156
  vs. functions, 155–156
  portfolio decision making and, 91–92
  portfolio optimization and, 84–85
  resource constraints, 89–91
  software mistake proofing, 301–303
Context-driven testing, 359
Continuous (and ordinal) critical parameters
  vs. discrete critical parameters, 238–241
  methods for deriving transfer functions from, 244–245
  MSA (measurement system analysis) and, 307
Contractual obligations, complications impacting prioritization, 93
Controls, for potential causes of failure in FMEA, 162–163
Cost-benefit analysis method (CBAM), 232–234
Counterproductive detractors, in portfolio decision making, 91
Cp/Cpk indices. *See also* Design capability; Process capability
  applying Six Sigma to process goals, 5
  calculating values for, 267–269
  capability analysis and, 202–203
  cooptimizing Cpk's, 282–283
  determining process capability with, 315–316
  difference between Cp and Cpk indices, 271–273
  forecasting/predictive use of, 265, 270
  M/PCpS (Machine/Process Capability Study), 6
  variance reduction and, 273–274
CPM, planning deployment and, 39
Creational design patterns, 354–355
Critical chain (theory of constraints) project management
  critical paths compared with, 98–102
  overview of, 103–105
Critical customer requirements (CCRs), 119
Critical parameter flow-down. *See also* Requirements flow-down
  examples, 206–208
  model for managing, 390
  overview of, 203–206
Critical parameter flow-up
  model for managing, 390
  Monte Carlo simulation of, 266–267
  overview of, 263–266
Critical Parameter Management
  DFSS goals and, 49
  formal gate reviews and, 56
Critical parameter scorecard, 269–270
Critical parameters
  constraints and, 156
  decision-making for software architecture and, 225
  definition of, 153–155

discrete vs. continuous, 238–241
examples of, 174–176
feature sets based on, 213–214
flow-down, 203–206
flow-down examples, 206–208
flow-up, 263–266
identifying, 49
models for managing flow-down and flow-up, 390
Monte Carlo simulation of flow-up, 266–267
position within DFSS flow, 153
predictive engineering and, 237
prioritization and selection of, 157–160
project schedule treated as, 95–96
RADIOV Architecture phase and, 75, 78
reliability and availability and, 172–174
requirements flow-down. *See* Requirements flow-down
target values and specification limits for, 197–198
VOB (voice of business) considerations, 155–156
Critical paths
critical chains compared with, 98–102
product development delays due to wandering critical path, 96
Crystal Ball
Monte Carlo simulation with, 84
portfolio decision making and, 89–90
Current Reality Tree, project selection and prioritization and, 37
Customer requirements
100-point rating method, 125
building House of Quality, 128–129
CCRs (critical customer requirements), 119
identifying challenging, 120–121
Kano model of, 122
translating into system requirements, 124–128
validation and prioritization of, 124
Customers. *See also* VOC (voice of customer)
critical parameters for optimization based on expectations of, 270
critical parameters impact on satisfaction of, 153
interview guide for, 113–115
interview process, 116
planning visits and interviews, 115–116
profile matrices, 111–112
reasons for failures experienced by, 313
reliability and availability expectations of, 172–174
reliability and availability perspective of, 319–321
retention impacting prioritization, 93
VOC gathering and, 111–112
voices and images in interview process, 112–113

D

DACE (Design and Analysis of Computer Experiments)
DOE compared with, 259–260
steps in, 260
Data collection plan, for Motorola ViewHome project, 219–220
Decision making
product portfolio. *See* Portfolio decision making
software architecture design, 224–227
Decomposition, 203–206. *See also* Flow-down/flow-up
Decomposition design heuristic, for software architecture, 228
Defaults, software mistake proofing and, 303
Defect discovery rate collection plan, 303–305
Defects
applying Six Sigma and, 5
binary logistic regression combined with DOE or RSM to minimize, 389
financial benefits of early detection, 10–11
software FMEA for detection of, 168
software stability and, 303–305
software testing for reducing, 347
testing as means of locating, 350

Define, Measure, Analyze, Design, Optimize, and Verify. *See* DMADOV (Define, Measure, Analyze, Design, Optimize, and Verify)
Define, Measure, Analyze, Improve, and Control. *See* DMAIC (Define, Measure, Analyze, Improve, and Control)
Delays
allocation of tolerances, 209
product development and, 96–98
Delighter's
assessing importance of, 157
in Kano model of customer requirements, 122–123
Deliverables
formal gate reviews and, 56
verification of supply chain readiness and, 363–364
Delivery. *See* On-time delivery
Demand uncertainty, supply chain decisions and, 372–373
Deploying DFSS
goals for DFSS and, 48–49
ideal scenario for, 29–30
overview of, 29
single project approach, 45–47
step 1: "burning platform", 30–32
step 2: guiding coalition, 32
step 3: defining the vision, 33–34
step 4: analyzing issues, 34–39
step 5: planning deployment, 39–40
step 6: communicating the vision, 40–41
step 7: executing deployment campaign, 41–42
step 8: removing impediments, 42–43
step 9: generating short-term wins, 43–44
step 10: consolidating gains, 44–45
success of, 50
summary, 50–51
tool set for, 47–48
Deployment experts, 29
Design
axiomatic, 145
of experiments. *See* DOE (design of experiments)
fractional factorial, 254–257
making insensitive to noise, 147
measuring agile, 218–219
robust concepts, 146–147
software architecture, 227–228, 234–235, 354
system, 187–190
Design and Analysis of Computer Experiments (DACE)
DOE compared with, 259–260
steps in, 260
Design capability. *See also* Cp/Cpk indices
assessment of, 202–203
calculating values for, 267–269
forecasting and, 270
predictive use of capability indices, 265
verification of. *See* Verification of design capability
Design FMEA. *See* DFMEA (Design FMEA)
Design for manufacturability and assembly. *See* DFMA (Design for manufacturability and assembly)
Design for Six Sigma. *See* DFSS (Design for Six Sigma) overview
Design heuristics, for software architecture, 227–228
Design patterns
benefits of, 355
example applications of, 355–356
GoF (Gang of Four), 354–355
software design and, 234–235, 354
Design phase, RADIOV, 78, 79
Detection ratings, for potential causes in FEMA, 162
Development manager (DM), risk assessment role of, 61, 352
DFMA (design for manufacturability and assembly), 366–369
best practices and benefits of, 369
list of key aspects and practices, 367–368
overview of, 366

DFMEA (Design FMEA). *See also* FMEA (Failure Modes and Effects Analysis)
for anticipation of problems, 194
benefits of, 196–197
in deployment planning, 39
DFSS goals and, 49
overview of, 195
tool summary, 196
DFSS (Design for Six Sigma) overview
Architecture phase, 17–18
CDMA base station example, 14–26
charter for IP BSC, 15–16
deployment. *See* Deploying DFSS
flowchart, 11, 364–365
history of, 8–9
Integration and Optimization phases, 18–21
key tools and methods, 12–13
preliminary steps in DFSS projects, 15
processes in, 9–11
Requirements phase, 15, 17
software DFSS. *See* SDFSS (software DFSS)
Verification phase of, 21–24
DFSS project Manager (DPM), 60
Diagnostic criteria, for success metrics, 57
Direction of goodness, in building House of Quality, 131
Discount rates, metrics for portfolio decision making, 88
Discrete critical parameters
vs. continuous critical parameters, 238–241
logistic regression for, 242–244
methods for deriving transfer functions from, 241–242
Distractions, portfolio decision making and, 91–92
DM (development manager), risk assessment role of, 61, 352
DMADOV (Define, Measure, Analyze, Design, Optimize, and Verify)
critical parameter flow-up and, 263–265
DFSS process nomenclature and, 69–70
DFSS steps, tools, and methods associated with, 12–13, 71–72
DOE (design of experiments) and, 251
in GE's DFSS project, 9
identification of critical parameters and, 153
RADIOV phases corresponding to, 15
requirements flow-down and, 187
schedule risks and, 95
software verification testing and, 350
verification of design capability and, 307
verification of reliability and, 319
DMAIC (Define, Measure, Analyze, Improve, and Control)
capability analysis and, 202
generating short-term wins, 43–44
identification of critical parameters, 153
overview of, 6–7
for problem-solving aspect of Six Sigma, 69
DOE (design of experiments), 251–256
applying to call processing failures in CDMA cell phone example, 22
benefits of, 251–252
DACE compared with, 259–260
fractional factorial design, 254–256
logistic regression combined with, 244–245, 389
measurement error and, 310
sparcity of effects principle in, 253–254
statistical methods for improving quality, 1–2
DPM (DFSS project Manager), 60

E

Early life failures
product tolerances, 4
verification of reliability and availability and, 326
Weibull distribution and, 323
Economic commercial value (ECV), 88–89
ECV (economic commercial value), 88–89
Electrical engineering equations, 246

Electrical tolerances, allocating, 209
Electronics products, applying DFSS to, 66–69
Empirical modeling
  using DOE, 251–256
  using historical data, 247–251
  using response surface methods, 256–259
End-to-end software testing
  definition of, 359–360
  overview of, 348–349
Entry/exit coverage, in end-to-end testing, 349
Equations
  for Cp and Cpk, 315
  electrical engineering and mechanical engineering, 246
  for modeling, 244–245
Errors
  preventing manufacturing and assembly errors, 366, 369
  preventing software mistakes and errors, 299–303
  sources of measurement error, 309–310
Event-driven reviews, 54
Excel, for modeling within spreadsheets, 246
Executing deployment campaign, steps in DFSS deployment, 41–42
Exploratory testing, 360

F

FACT TOPS Team, 9, 274, 283–288
Failover testing, 360
Failure modes. *See also* FMEA (Failure Modes and Effects Analysis)
  benefits of FMEA in anticipating, 195–197
  benefits of P-diagrams in anticipating, 195
  determining risk of failure, 315–316
  listing in FMEA, 160
  requirements flow-down process and, 193–194
Failure Modes and Effects Analysis. *See* FMEA (Failure Modes and Effects Analysis)
Failures
  acceleration factors, 329–330
  early life failures/infant mortality, 326
  list of common software failures, 353
  risk of failures despite verification, 331–332
  useful life/constant failure rates, 325–326
  wear out mechanisms, 326
  WeiBayes approach to, 330–331
Fault tree analysis. *See* FTA (fault tree analysis)
Feasibility screening, in concept generation/selection, 148–149
Feature development, iterative approach to, 213–214
Feedback
  governance as feedback mechanism, 53
  software mistake proofing and, 301
Feldbaumer, David, 9
Field testing, 360
Fiero, Janet, 3
Financial metrics, for portfolio decision making, 85–89
Flow-down/flow-up
  availability and reliability and, 321–322
  criteria for success metrics, 57
  critical parameter flow-down. *See* Critical parameter flow-down
  critical parameter flow-up. *See* Critical parameter flow-up
  requirements flow-down. *See* Requirements flow-down
FMEA (Failure Modes and Effects Analysis)
  design FMEA for anticipation of problems, 194–197
  design FMEA for deployment planning, 39
  DFSS goals and, 49
  evaluating system-level risks, 157–158
  formal gate reviews and, 56
  risk reduction and, 353
  software FMEA acronyms and definitions, 164
  software FMEA benefits, 168–169
  software FMEA cost savings and ROI, 167–168
  software FMEA implementation case study, 169–172

software FMEA process documentation, 176–185
software FMEA process phases, 165–167
software FMEA roles and responsibilities, 165
system FEMA, steps in, 160–163
system FMEA risk evaluation, 157–158
Forums, for communicating vision, 41
Fractional factorial design
CCD (central composite design) based on, 256–257
DOE (design of experiments) and, 254–255
FTA (fault tree analysis)
applying to CDMA example, 19
DFSS goals and, 49
of reliability, 173–174
Fullerton, Craig, 9
Functional modeling, for hardware, 144
Functional testing, 360
Functions/functionality
decision-making for software architecture and, 225
end-to-end testing and, 349
functions vs. constraints, 155–156
interface functionality, 369
listing functions in FEMA, 160
measurable requirements and, 156
software testing and, 347, 350
transfer functions. *See* Transfer functions
Future directions
Agile approach to change management, 389
DFSS integration with systems engineering, 388
innovation, 388–389
logistic regression for minimizing defects, 389
modeling flow-down and flow-up of critical parameters (Otto), 390
portfolio including services as well as projects, 388
project and program management, 388
reliability modeling, 390
risk management process (Mun), 387–388
strategies becoming tactics, metrics, and actions, 386–387
Future Reality Tree, 38
"Fuzzy front end", causing product development delays, 97–98

G

Galvin, Bob
Malcolm Baldrige National Quality Award, 7–8
role in development of Six Sigma, 5
Garvin, David, 108–109
Gate reviews, for governance, 55–56
Gauge repeatability and reproducibility. *See* GR&R (gauge repeatability and reproducibility) index
General Electric History of Six Sigma and, 2, 9
General linear model (GLM), 247–251
Generation of system moments
critical parameter scorecard, 269–270
predicting critical parameter values, 267–269
GLM (general linear model), 247–251
Goals
for DFSS deployment, 48–49
portfolio decision making and, 91–92
Governance, 53–56
formal gate reviews, 55–56
overview of, 53–54
supportive project reviews, 54–55
GR&R (gauge repeatability and reproducibility) index
improving inadequate measurement systems, 312
MSA and, 202, 309–310
Green belts
certification of, 62–63
risk assessment role of, 352
"Guard-banding" approach, to measurement error, 314
Guiding coalition
for DFSS deployment, 32
role in removing roadblocks and impediments, 42

H

Hardware
alternate architecture generation for, 143–146
applying DFSS to, 66–67
RADIOV and, 11
requirements flow-down for, 190–193
tolerance expectations, 366
Harry, Dr. Mikel
documentation of Six Sigma concepts, 6
as head of SSRI, 8
Heuristics (rule of thumb)
design heuristics for software architecture, 227–228
for generating alternative concepts and architectures, 145–146
Higher-is-better critical parameters, 197
Historical data, empirical modeling from, 247–251
House of Quality
competitive analysis and, 129
constructing, 128
critical parameter flow-down or decomposition, 203
customer requirements, 128–129
direction of goodness, 131
as method to flow-down requirements, 191–192
prioritization and, 129, 133
relationship matrix, 131–132
system-level critical parameters, 190–191
system requirements, 129–131
targets and units for system requirements, 133
trade-offs among system requirements and, 132
translating customer requirements into system requirements, 127–128

I

Iacocca, Lee, 108
IAR (integrated alternator regulator), predictive engineering case study, 288–290
ICs (integrated circuits), statistical methods for improving quality of, 2
Identification
of challenging customer requirements, 120–121
of critical parameters, 49, 153
in risk management, 60
IDOV (Identify, Design, Optimize, Verify)
DFSS process nomenclature, 69–70
DFSS project phases, 9
DFSS steps, tools, and methods associated with, 71–72
RADIOV phases corresponding to, 15
Images
KJ analysis and, 117–118
VOC gathering and, 112–113
Impediments. *See* Barriers, to acceptance of DFSS
Incremental integration testing, 360
Indifferent's, in Kano model, 122–123
Infant mortality, reliability and availability and, 326
Initial tolerance allocation, in requirements flow-down, 208–210
Innovation, future directions for applying DFSS, 388–389
Instability, of requirements, 59
Install/uninstall testing, 360
Integer programming, portfolio decision making and, 89
Integrate phase, RADIOV, 78, 79
in CDMA cellular phone example, 18–21
Integrated alternator regulator (IAR), predictive engineering case study, 288–290
Integrated circuits (ICs), statistical methods for improving quality of, 2
Inter-operability testing (IOT), 348
Interfaces
appropriateness and acceptability of flows between, 369
software mistake proofing and, 303

Interruptions, product development delays due to, 101
Interviewing customers
guide for, 113–115
planning, 115–116
process of, 116
Inventory, in supply chain decision-making, 372, 375, 378
IOT (inter-operability testing), 348
IP-BSC (IP-based base station controller)
aligning critical parameters with DFSS tools, 19
benefits of DFSS project for, 25–26
CDMA base station example and, 14–16
Iridium project, Motorola, 107
Issues, analyzing. *See* Analyzing concerns and issues
Iterative development
Agile development and, 212–213
DFSS tools used with, 215–217
schedule risks and, 105–106

J

Japan Society of Kansei Engineering (JSKE), 152
Job titles and responsibilities, as impediment to acceptance of DFSS, 38–39
JSKE (Japan Society of Kansei Engineering), 152

K

"k" factor, in Cp and Cpk capability indices, 271–273
Kano, Dr. Noriaki, 122
Kano model, 122–123
Kansei engineering, 152
Kawakita, Jiro, 117
Key performance indicators (KPIs), 294–295, 297
KJ (Jiro Kawakita) analysis
affinity diagramming compared with, 114
overview of, 117–120
risk management and, 48
"Knapsack problem", portfolio optimization and, 84–85
Kotter, Dr. John, 30
Kougaku, Kansei, 152
KPIs (key performance indicators), 294–295, 297

L

Launch schedules, products, 369–370
Lawson, Dr. J. Ronald
documentation of Six Sigma concepts, 6
in history of Six Sigma and DFSS, 2
Layers, software architecture, 223–224
Lead time
supply chain readiness and, 370–372
trade-offs between on-time delivery, lead time, and inventory levels, 372, 380
Leadership
consolidating gains and, 44
by example, 41
ideal scenario for DFSS deployment, 30
role in removing roadblocks and impediments, 43
Leading Change (Kotter), 30
Leading indicators, criteria for success metrics, 57
Lean development
Agile development compared with, 211
early example of, 2
Legacy architecture, risks of changing, 230
Linear regression, empirical modeling using historical data, 247–251
Linear Satisfier's, in Kano model, 122–123
Load tests
definition of, 360
software testing and, 347
Logistic regression
for discrete critical parameters, 242–244
DOE combined with, 244–245
minimizing defects, 389
"Loss leaders", financial metrics for portfolio decision making, 85
Lower-is-better critical parameters, 197

M

M/PCpS (Machine/Process Capability Study), 6
Maass, Eric
  FACT TOPS Team, 9, 274, 283-288
  in history of Six Sigma and DFSS, 2
Machine/Process Capability Study (M/PCpS), 6
Malcolm Baldrige National Quality Award, 7
Management
  DFSS goals and, 48
  improving project and program management, 388
  role in removing roadblocks and impediments, 43
Manufacturability
  DFMA (design for manufacturability and assembly), 366–369
  tools for reviewing, 49
Market penetration, complications impacting prioritization, 93
Marketing
  customer profile matrices and, 111–112
  VOC gathering and, 107–108
Mathematical modeling software, 246
MBB (Master Black Belt)
  certification by, 62–63
  risk management and, 61
MCM (multichip module), supply chain readiness case study, 380–382
Mean
  generation of system moments for predicting values of, 267
  in optimization of critical parameters, 271–273
Mean time between failures (MTBF), 320–321
Mean-time-to-failure (MTTF)
  critical parameters and, 197
  reliability and availability and, 320–321
Measurable requirements
  Agile development and, 218–219
  critical parameters as, 153
  data collection plan for ViewHome project, 219–220
  functionality and, 156
Measurement error
  "guard-banding" approach to, 314
  MSA and, 199
  sources of, 309–310
Measurement phase, DFSS, 218
Measurement system analysis. *See* MSA (measurement system analysis)
Measurement systems
  averaging measurements to improve, 312
  improving inadequate, 310–311
Mechanical engineering equations, 246
Mechanical tolerances, allocating, 209
Media, for communicating organizational vision, 41
Metrics. *See also* Success metrics
  applying Six Sigma to process goals, 5
  defining, 41–42
  key performance, 297
  portfolio decision making and, 85–89
  risks and, 59
MICARL (Motorola Integrated Circuits Applications Research Laboratory), 1
Microsoft Project, 104
Middleware, in software architecture, 224
Mind-mapping, for concept generation, 140–141
Minitab reliability tools, 330–331
Modeling
  availability, 342–346
  critical parameter flow-down/flow-up, 390
  DACE and, 259–260
  empirical modeling using DOE, 251–256
  empirical modeling using historical data, 247–251
  empirical modeling using RSM, 256–259
  evaluating software optimization models, 298–299
  existing or derived equations for, 245–246
  reliability, 390
  schedule risks, 95–96
  software architecture design, 234–235
  software options for, 246–247

supply chain decisions in optoelectronic multichip module, 380–382
supply chain decisions in semiconductor manufacturing, 372–379
Modifiability tactic, in software architecture design, 229
Modules
concept generation at module level, 139
interface flows and, 369
Monte Carlo simulation
for cooptimizing Cpk's, 283
of critical parameter flow-up, 266–267
critical parameter scorecard and, 269–270
for DFSS goals, 49
history of development of, 266
for optimizing guard-bands, 314–315
for overcoming resistance to change, 32
for product or service evaluation, 83–84
for schedule estimation, 99–100
of selling price resulting in profit or losses, 86–87
showing impact of Parkinson's Law on project scheduling, 102
software options for modeling, 247
of system availability, 21, 23
use case modeling and, 298
MotoOATSGen tool, 356
Motorola
history of Six Sigma and DFSS, 1–2
Iridium project, 107
Malcolm Baldrige National Quality Award, 7–8
TCS (total customer satisfaction) competition, 8
Motorola Integrated Circuits Applications Research Laboratory (MICARL), 1
Motorola Training and Education Center (MTEC), 3
MSA (measurement system analysis), 198–202
linking to test and verification phase, 201–202
measurement error and, 199
overview of, 198–199
performing on measurable requirements, 75
variance and, 200–201, 272
verifying design capability, 307–310
MTBF (mean time between failures), 320–321
MTEC (Motorola Training and Education Center), 3
MTTF (mean-time-to-failure)
critical parameters and, 197
reliability and availability and, 320–321
Multichip module (MCM), supply chain readiness case study, 380–382
Multiple response optimization
overview of, 280–282
software performance and, 293–294
YSM (Yield Surface Modeling) and, 283–288
Multitasking
critical chain project management and, 100–102
minimizing, 104
product development delays due to, 96
Mun, Dr. Johnathan, 387–388
Must-Be's, in Kano model, 122–123
Mutation testing, 360

N

National Institute of Standards and Technology (NIST), 309
Natural mapping, software mistake proofing, 299–300
Net present value (NPV) metric, for portfolio decision making, 88–89
New product introduction (NPI), 58–59
New-unique-difficult (NUDs), customer requirements, 120–121
NIH ("Not Invented Here") syndrome, 32
NIST (National Institute of Standards and Technology), 309
Noise
Agile development and, 217–218
making design insensitive to, 147
"Not Invented Here" (NIH) syndrome, 32
NPI (new product introdution), 58–59

NPV (net present value) metric, for portfolio decision making, 88–89
NUDs (new-unique-difficult), customer requirements, 120–121
Numbers, software mistake proofing, 303

O

OATS (Orthogonal-array based testing), 356–358, 360
Objectives, of VOC gathering, 110
Occurrence ratings, assigning to causes of failure, 161–162
On-time delivery
  supply chain readiness and, 370–372
  trade-offs between on-time delivery, lead time, and inventory levels, 372
Ooi, C.C., 283
Open-ended questions, in interviewing, 114
Operating systems (OSs), 223–224
Optimization
  combinatorial optimization algorithms, 356–357
  cooptimizing Cpk's, 282–283
  mean and/or variance in, 271–273
  multiple response optimization, 280–282
  portfolio decision making as optimization process, 84–85
  robustness achieved through variance reduction, 273–280
  selecting critical parameters for, 270
  software optimization. *See* Software optimization
Optimize phase, RADIOV
  in CDMA cellular phone example, 18–21
  overview of, 78–80
  tools, 80
Optobus module, 380
Optoelectronic multichip module (MCM), supply chain readiness case study, 380–382
OptQuest
  cooptimizing Cpk's, 283
  portfolio decision making and, 89–91
Oracle, Crystal Ball utility, 84
Ordinal critical parameters. *See* Continuous (ordinal) critical parameters
Orthogonal-array based testing (OATS), 356–358, 360
Orthogonality criteria, for success metrics, 57
OSs (operating systems), 223–224
Otto, Dr. Kevin, 390

P

P-diagrams
  anticipation of potential problems, 195
  applying DFSS tools to Agile development, 216–217
  DFSS goals and, 49
  for requirements flow-down, 191
P/T (precision-to-tolerance) ratio
  improving inadequate measurement systems, 312
  MSA and, 202, 309–310
  system analysis and, 197
P × I × T (Probability × Impact × Time frame), risk formula, 61–62
Parameters, critical. *See* Critical parameters
Pareto analysis, of cause categories, 171
Parkinson's Law
  critical chain project management and, 100–101
  Monte Carlo simulation showing, 102
  product development delays due to, 96
Path coverage, end-to-end testing and, 349
Payback time metric, for portfolio decision making, 85–89
PDMA (Product Development and Management Association), 107
Perez-Wilson, Mario, 6
Performance
  metrics for, 294–295, 297
  as quality attribute, 227
  software architecture design tactics, 229
  software systems and, 293–294
  software testing and, 347
  tools for measuring robustness of, 49

Performance testing, 347, 360
Personal buffers, in critical chain project management, 104
Phases, DFSS, 64
Pilot runs, manufacturability and, 49
Planning
  customer visits and interviews, 115–116
  DFSS deployment, 39–40, 47
  product launch, 370
  risk management and, 60, 351–352
  RPN reduction, 163
Planning backwards, in critical chain project management, 103
PM (Project manager), 60, 352–353
Poka Yoke, 369
Poppendieck, Mary, 211
Poppendieck, Tom, 211
Portfolio decision making
  business case risk and, 94
  financial metrics and, 85–89
  goals, constraints, considerations, distractions, 91–92
  impact of commitments and strategic direction on, 92–94
  as optimization process, 84–85
  overview of, 83–84
  resource constraints and, 89–91
  steps in analyzing project already underway, 93–94
Pp/Ppk. *See* Cp/Cpk indices
Precision
  measurement variability and, 310
  MSA and, 309
Precision-to-tolerance ratio. *See* P/T (precision-to-tolerance) ratio
Prediction criteria, for success metrics, 57
Prediction indices, for capability, 265
Predictive engineering
  cooptimizing Cpk's, 282–283
  critical parameter flow-up and, 263–266
  critical parameter scorecard, 269–270
  DACE and, 259–260
  deriving transfer functions for continuous critical parameters, 244–245
  deriving transfer functions for discrete critical parameters, 241–242
  discrete vs. continuous critical parameters, 238–241
  empirical modeling using DOE, 251–256
  empirical modeling using historical data, 247–251
  empirical modeling using response surface methods, 256–259
  existing or derived equations for modeling, 245–246
  future directions for applying DFSS, 389
  generation of system moments, 267–269
  IAR (integrated alternator regulator) case study, 288–290
  logistic regression for discrete parameters, 242–244
  mean and/or variance in optimization of critical parameters, 271–273
  Monte Carlo simulation of critical parameter flow-up, 266–267
  multiple response optimization, 280–282
  optimizing robustness through variance reduction, 273–280
  overview of, 237–238
  selecting critical parameters for optimization, 270
  software optimization. *See* Software optimization
  software options for modeling, 246–247
  summary, 261, 290–291
  YSM and, 283–288
Prioritization
  building House of Quality, 129
  complications in, 92–93
  of critical parameters, 157–160
  of customer requirements, 124
  of resources, 36–38
  of system requirements, 133
  trade-off analysis with Prioritization matrix, 232

Privileges, software mistake proofing and, 303
Probability × Impact × Time frame (P × I × T), risk formula, 61–62
Problem anticipation. *See* Failure modes
Process capability. *See also* Cp/Cpk indices
  applying Six Sigma to process goals, 5
  determining, 315–316
  M/PCpS (Machine/Process Capability Study), 6
Product development
  applying DFSS to, 65
  benefits of, 385–386
  existing process as impediment to acceptance of DFSS, 34–35
  factors in product development time, 96
  innovation and, 389
  reasons for cancellation or failure, 107
  risks, 58–60
  schedule risks. *See* Schedule risks
Product Development and Management Association (PDMA), 107
Product launch schedules, 369–370
Products
  applying VOC to, 48
  assembly. *See* DFMA (design for manufacturability and assembly)
  confidence in meeting on-time delivery, 370
  metrics for delivery risks, 59
  portfolio decision making. *See* Portfolio decision making
  roadmaps for product portfolio, 387
Profit metric, for portfolio decision making, 85–89
Programs, improving management of, 388
Project buffers, in critical chain project management, 105
Project manager (PM), 60, 352–353
Project reviews, for governance, 54–55
Projects
  improving management of, 388
  removing doomed, 84
  schedule risks. *See* Schedule risks
  single project approach to DFSS deployment, 45–47
Proofing, for software mistakes and errors, 299–303
Prototyping, in software architecture design, 234–235
Pugh concept selection process
  DFSS goals and, 49
  overview of, 149–151
Pugh, Dr. Stuart, 149–151
Pugh matrix, trade-off analysis with, 231

Q

Q × A = E (quality × acceptance = effectiveness), 40
QFD (quality function deployment)
  concept selection process and, 110
  planning deployment and, 39
Qualifications, iterative development and, 105–106
Quality attributes, for software architecture systems, 225–227
Quality function deployment (QFD)
  concept selection process and, 110
  planning deployment and, 39
Quality × acceptance = effectiveness (Q × A = E), 40

R

RADIOV (Requirements, Architecture, Design, Integration, Optimization, and Verification)
  alignment with DFSS flow, 75
  Architecture phase, 17–18, 75, 77–78
  capability analysis and, 202
  critical parameter flow-up and, 263–265
  Design phase, 78
  DFSS process nomenclature and, 69–70
  DFSS steps, tools, and methods associated with, 12–13, 71–72
  DOE and, 251

Integrate phase, 18–21, 78
merging DFSS hardware and software, 11
Optimize phase, 18–21, 78–80
requirements flow-down and, 187
Requirements phase, 15, 17, 73–74, 76
SDFSS methods and, 215
software verification testing and, 350
TDD and, 212
trade-off analysis tools, 231
verification of design capability and, 307
verification of reliability and, 319
Verify phase, 21–24, 80–81
Rayleigh model, 304–305
Recognition, consolidating gains in DFSS deployment, 44
Recovery testing, 360
Red X effect, 274
Regression analysis, 247–251
Regression testing
binary logistic regression, 389
definition of, 360
software testing and, 348
Relationship matrix, in building House of Quality, 131–132
Release capability, predicting for software, 304–305
Release to product, schedule risks and, 105–106
Reliability
bathtub curve for interval aspect of, 323
critical parameters and, 172–174
customer perspective on, 320
definition of, 322
flow-down, 321–322
metrics for reliability risks, 59
modeling, 49, 390
of software, 325
software reliability case study, 333–341
verifying. *See* Verification of reliability and availability
Removing impediments, steps in DFSS deployment, 42–43
Replication design heuristic, for software architecture, 228
Reproducibility, measurement variability and, 311
Requirements
Agile handling of changes in, 389
business requirements, 110
customer requirements. *See* Customer requirements
gathering (VOC and VOB), 214
impact of changing requirements on scheduling, 97–98
instability of, 59
system requirements. *See* System requirements
tolerance expectations for hardware requirements, 366
Requirements, Architecture, Design, Integration, Optimization, and Verification. *See* RADIOV (Requirements, Architecture, Design, Integration, Optimization, and Verification)
Requirements flow-down
anticipation of potential problems, 193–194
benefits of FMEA in anticipation of problems, 195–197
benefits of P-diagrams in anticipation of problems, 195
capability analysis and, 202–203
critical parameter flow-down or decomposition, 203–206
flow-down examples, 206–208
for hardware and software systems, 190–193
initial tolerance allocations and, 208–210
MSA and, 198–202
position within DFSS flow, 187–190
summary, 210
target values and specification limits for critical parameters, 197–198
Requirements phase, RADIOV
in CDMA cellular phone example, 15, 17
overview of, 73–74, 76

Resolution, risk management and, 60
Resource sharing design heuristic, for software architecture, 228
Resources
critical chain (theory of constraints) project management and, 104–105
dependencies in critical chain project management, 101
ideal scenario for DFSS deployment, 30
impact of constraints on portfolio decision making, 89–91
resource requirement as impediment to acceptance of DFSS, 36–38
Response surface methods. *See* RSM (response surface methods)
Responsibilities
impediments to acceptance of DFSS, 38–39
in software FMEA, 165
Return on investment. *See* ROI (return on investment)
Revenue metric, for portfolio decision making, 85–86
Review process
formal gate reviews, 55–56
Rigorous Gate Reviews, 48
supportive project reviews, 54–55
Rewards, consolidating gains and, 44
Rigorous Gate Reviews, 48
Risk management
DFSS goals and, 48
Mun proposal for, 387–388
planning, 351–352
Risk management roles
development manager, 61, 352
overview of, 60
project manager, 60, 352–353
risk owner, 61–62, 352–353
Risk of failures
despite verification, 331–332
verification of design capability and, 313–315
Risk owner, 61–62, 352–353
Risk priority numbers. *See* RPNs (risk priority numbers)
Risks
business case. *See* Business case risks
existing solutions and, 147
FMEA for evaluating system-level, 157–158
plans for reducing RPNs, 163
prioritization and selection of critical parameters and, 157
product development, 58–60
schedule. *See* Schedule risks
stakeholders assessing, 159–160
steps in assessing, 352
success metrics and, 155–156
trade-off analysis for assessing, 230
Roadblocks, analyzing potential, 34
Robust design
concepts, 146–147
DFMA and, 366–369
software verification testing and, 350
variance reduction and, 273–280
ROI (return on investment)
CBAM (cost-benefit analysis method) and, 233–234
financial metrics for portfolio decision making, 85–89
software FMEA and, 167–168
Roles
development manager, 61, 352
project manager, 60, 352–353
risk owner, 61–62, 352–353
software FMEA, 165
use case modeling and, 295
RPNs (risk priority numbers)
calculating and applying in FEMA, 162–163
formal gate reviews and, 56
formula for determining, 61–62
system availability and, 18

RSM (response surface methods)
benefits of combining with binary logistic regression, 389
cooptimizing Cpk's, 283
empirical modeling, 256–259
multiple response optimization, 280–282
YSM (Yield Surface Modeling) compared with, 283–284
Rule of thumb. *See* Heuristics (rule of thumb)

S

Sanity testing, 361
Satisfaction, in Kano model, 122
Schedule risks
allocation of tolerances, 209
changing requirements and, 97–98
critical chain theory of constraints project management and, 103–105
critical paths vs. critical chains in determining durations, 98–102
iterations, qualification, and release to product, 105–106
metrics for, 59
model for, 95–96
position within DFSS flow, 95
summary of, 106
Schedules, product launch, 369–370
Scripting languages, 224
SDFSS (software DFSS). *See also* Agile development
applying DFSS to software development, 67
combining with Agile development, 211
DFSS tools used with Agile development, 215–217
in Motorola ViewHome project, 219–220
SDLC (software development lifecycle)
Agile development and, 212
DFSS supporting, 214
SDM (systems development manager), 61
Security tactics, in software architecture design, 229–230
Security testing, 348, 361
Semiconductor manufacturing, product delivery example, 372–379
"Sense engineering", 152
Sensitivity studies, model evaluation and, 298
Separation design heuristic, for software architecture, 228
Services
application support services in software architecture design, 224
applying DFSS to, 65–66
applying VOC to, 48
portfolio of projects including, 388
system services in software architecture design, 223–224
Severity ratings, assigning to potential failure modes, 160–161
Shainin, Dorian, 3–4, 274
Short-term wins, generating in DFSS deployment, 43–44
Simulation
DACE and, 259–260
modeling with simulation software, 246
Monte Carlo simulation. *See* Monte Carlo simulation
software architecture design and, 234–235
SPICE simulation, 285–286, 289
SUPREM for process simulation, 2
YSM and, 283
Single project approach, to DFSS deployment, 45–47
Six Sigma
background of, 1–3
Bill Smith's role in birth of, 3–5
Cp and Cpk indices for, 315
preference for continuous parameters in, 238
six steps to, 6–7
verification test strategy using, 350–354
Six Sigma Design Methodology (SSDM), 9
Six Sigma Research Institute (SSRI), 8

Smith, Bill, 3–5
Smoke testing, 361
Software
development lifecycle. *See* SDLC (software development lifecycle)
execution models, 298
for modeling, 246–247
RADIOV and, 11, 78
reliability, 325
reliability case study, 333–341
requirements flow-down, 190–193
Software architecture
alternate architecture generation for, 143–146
decision-making process for, 224–227
design heuristics for, 227–228
design patterns, simulation, modeling, and prototyping, 234–235
flowchart for decision-making, 226
layers, 223–224
overview of, 223
summary, 235
tactics, 228–230
trade-off analysis, 230–234
Software DFSS. *See* SDFSS (software DFSS)
Software FMEA. *See also* FMEA (Failure Modes and Effects Analysis)
acronyms and definitions, 164
benefits of, 168–169
cost savings from, 167–168
implementation case study, 169–172
presentation template, 168
process documentation, 176–185
process phases, 165–167
roles in, 165
tracker, 167
Software optimization
model evaluation, 298–299
multiple response optimization, 293–294
overview of, 293
proofing for mistakes and errors, 299–303
stability, 303–305
summary, 305
use case modeling, 294–298
Software testing
list of common software failures, 353
overview of, 347–350
summary, 358–359
terminology related to, 359–361
test case development with design patterns, 354–356
types of, 359–361
verification test strategy using Six Sigma, 350–354
verification testing using combinatorial design method, 356–358
Sources of variability (SOV), 2, 272
SOV (sources of variability), 2, 272
Sparcity of effects principle, in statistics, 253–254
Special characters, software mistake proofing, 303
Specification limits, for critical parameters, 197–198
SPICE simulation
IAR (integrated alternator regulator) case study and, 289
YSM and, 285–286
Spreadsheets, modeling within, 246
Sprints, in Agile development, 213
SSDM (Six Sigma Design Methodology), 9
SSRI (Six Sigma Research Institute), 8
Stability
instability of requirements, 59
software optimization and, 303–305
Stakeholders
certification of, 63
engaging in DFSS deployment, 32
numerical assessment of risks by, 159–160
role in defining the vision, 33
single project approach to DFSS deployment, 46–47
Standard deviation, 267, 271–273

Stanford University Process Engineering Model (SUPREM), 2
Statistics
methods for improving quality, 1–3
sparcity of effects principle in, 253–254
Strategies
adjusting decisions based on, 92–94
portfolio decision making and, 91
tactics derived from, 386
Stress testing
interface functionality, 369
overview of, 361
software, 347
Strings, software mistake proofing, 303
Structural design patterns, 354–355
Subsystem level
concept generation at, 139
interface flows and, 369
system design and, 189–190
Success metrics
additive and multiplicative models for, 57–58
criteria for, 57
defining, 41–42
formal gate reviews and, 56
program risks and, 155–156
Supply chain readiness
interface flows and, 369
on-time delivery and lead time commitments and, 370–372
optoelectronic multichip module case study, 380–382
position within DFSS flow, 363–365
product assembly robustness and, 366–369
product launch schedules and, 369–370
semiconductor manufacturing example related to on-time delivery, 372–379
summary, 382–383
tolerance expectations and, 366
trade-offs between on-time delivery, lead time, and inventory levels, 372
SUPREM (Stanford University Process Engineering Model), 2
Symbols, in use case modeling, 295
System engineering, incorporating DFSS with, 388
System FMEA. *See also* FMEA (Failure Modes and Effects Analysis)
evaluating system-level risks, 157–158
steps in system/subsystem-level FEMA, 160–163
System level
concept generation at, 139
design process, 187–189
quality attributes for software architecture systems, 225–227
question to ask for system verification, 351
System requirements. *See also* Requirements flow-down
building House of Quality, 129–131
prioritization of, 133
process of system design and, 187–189
setting roof on trade-offs among, 132
targets and units for, 133
translating customer requirements into, 124–128
System services (layer 1), software architecture design, 223–224
System testing, 348, 361

T

Tactics
software architecture design, 228–230
strategies becoming, 386
Target-is-best critical parameters, 198
Target values, for critical parameters, 197–198
Taylor series expansion, 267–268
TCS (total customer satisfaction), Motorola competition, 8
TDD (test-driven development)
Agile development and, 218
overview of, 212

TDFSS (Technology Development for Six Sigma), 65
Team
role in defining the vision, 33–34
scenario for DFSS deployment, 30
stakeholder engagement in DFSS deployment, 32
support for, 44
VOC gathering, 110–111
Technical experts, compared with deployment experts, 29
Technical lead, 63–64
Technical requirements, 154, 157
Technical risks, metrics for, 59
Technology Development for Six Sigma (TDFSS), 65
Teoriya Resheniya Izobreatatelskikh Zadatch (TRIZ), 141–143
Test cases
design patterns for developing, 354–356
principals for developing, 349–350
Test-driven development (TDD)
Agile development and, 218
overview of, 212
Test escapes
reliability and, 331–332
verification of design capability and, 313–315
Testability tactics, in software architecture design, 229
Tests
accelerated life testing, 328–330
software. *See* Software testing
types of, 359–361
Theory of Innovative Problem Solving (TIPS). *See* TRIZ (Teoriya Resheniya Izobreatatelskikh Zadatch)
Thread-based testing, 361
Time requirements, as impediment to acceptance of DFS, 35–36
"Time trap", in Lean Six Sigma, 380
Timing/delay, allocation of tolerances, 209
TIPS (Theory of Innovative Problem Solving). *See* TRIZ (Teoriya Resheniya Izobreatatelskikh Zadatch)
Tolerances
initial tolerance allocation, 208–210
P/T (precision-to-tolerance) ratio. *See* P/T (precision-to-tolerance) ratio
in supply chain readiness, 366
Tools
associated with DFSS steps, 71–72
DFSS tools used with Agile development, 215–217
minimum toolset for deploying DFSS, 47–48
RADIOV, 77, 79–81
single project approach to DFSS deployment, 46
for trade-off analysis, 231
Total customer satisfaction (TCS), Motorola competition, 8
Trade-off analysis
CBAM (cost-benefit analysis method), 232–234
decision-making and, 224–225
prioritization matrix, 232
Pugh matrix, 231
software architecture design decisions, 230
Training
certification and, 62–63
DFSS deployment and, 39, 46
Transfer functions
in conjunction with estimated distributions, 264–265
definition of, 237
deriving from continuous critical parameters, 244–245
deriving from discrete critical parameters, 241–242
determining need for, 237–238
Monte Carlo simulation used with, 267
predicting performance with, 263
TRIZ Contradiction Matrix, 142–143

TRIZ (Teoriya Resheniya Izobreatatelskikh Zadatch), 141–143

U

UML (unified modeling language)
software development and, 144
use case modeling and, 296
Uncertainty, supply chain decisions and, 372
Unified modeling language (UML)
software development and, 144
use case modeling and, 296
Unit tests
definition of, 361
in software testing, 347
Usability tactics, in software architecture design, 229
Usability testing, 361
Use-based (cluster) testing, 361
Use case modeling, 294–298
Useful life, in verifying reliability and availability, 325–326
User acceptance testing, 361
User interface (layer 3), software architecture design, 224
Users
interactions via software applications, 224
interface flows and, 369

V

"V" model, 390
Validation
of customer requirements, 124
of inputs, in software mistake proofing, 302
question to ask for system validation planning, 351
Validation testing, 361
Variance
MSA and, 200–201
in optimization of critical parameters, 271–273
red X effect and, 274
reduction methods, 273–280
Six Sigma focus on variance reduction, 3
Verification
Agile development and, 218
data collection plan for ViewHome project, 220
linking MSA to, 201–202
software testing. *See* Software testing
of supply chain readiness. *See* Supply chain readiness
Verification of design capability
assessing capability, 315–316
improving measurement systems, 310–313
MSA and, 307–310
position within DFSS flow, 307
risk of failures despite, 313–315
summary, 316–317
Verification of reliability and availability
accelerated life testing, 328–330
availability and reliability flow-down, 321–322
bathtub curve and Weibull distribution and, 322–325
customer perspective on, 319–321
early life failures/infant mortality, 326
flowchart for, 327–328
methods for improving, 332
modeling availability case study, 342–346
risk of failures despite verification, 331–332
software reliability, 325
software reliability case study, 333–341
summary, 333
useful life/constant failure rates, 325–326
wear out mechanisms, 326
WeiBayes approach, 330–331
Verification testing
combinatorial design method for, 356–358
using Six Sigma, 350–354
Verify phase, DFSS, 307, 363
Verify phase, RADIOV
in CDMA base station example, 21–24
overview of, 80–81
tools, 81

ViewHome project (Motorola), 219–220
Visibility, software mistake proofing, 300
Vision (organizational)
  communicating, 40–41
  defining, 33–34
VOB (voice of business)
  business requirements, 110
  critical parameters and, 155–156
  requirements gathering, 214
VOC (voice of customer)
  applying to new products or services, 48–49
  critical parameters as response to, 153–155
  customer expectations regarding reliability and availability, 319–320
  customer interview guide, 113–115
  customer selection for, 111–112
  developing services in response to, 388
  DFSS aligned with, 9–10
  House of Quality and, 128–133
  identifying challenging customer requirements, 120–121
  implicit VOC, 155
  importance of, 107–108
  interviewing customers, 116
  Kano model, 122–123
  KJ analysis, 117–120
  planning customer visits and interviews, 115–116
  position in DFSS flow, 108–110
  purpose and objectives of, 110
  RADIOV requirements phase and, 73
  requirements gathering, 214
  Six Sigma steps related to, 6
  summary of, 134–135
  team for VOC gathering, 110–111
  translating customer requirements into system requirements, 124–128
  validation and prioritization of customer requirements, 124
  voices and images in interview process, 112–113
Voices
  KJ analysis and, 117–118
  VOC gathering and, 112–113

W

"W" model (Otto), 390
Waterfall model, 105
Wear out mechanisms, 325, 326
WeiBayes approach, 330–331
Weibull distribution, 322–325
  early life failures/infant mortality, 326
  shape and scale parameters, 323
  useful life/constant failure rates, 326–327
  wear out mechanisms, 327
  WeiBayes approach to failures, 330–331
White box tests
  definition of, 361
  in software testing, 347
Who Moved My Cheese? (Johnson and Blanchard), 30–31

Y

YSM (Yield Surface Modeling), 283–288