

OTHER BACK MATTER

*Notes (which provide detail without interrupting the flow),
Bibliography (which backs up the notes), Customer Council
(which advised us on content), and Index*

from

Flexible Product Development

by

Preston G. Smith

Notes

Introduction

1. *Iteration* comes from the software development world (see Chapter One). In the lean manufacturing and lean product development fields, the related term is *small batch size*, for which see Reinertsen's *Managing the Design Factory*.

Chapter One

1. See the Boston Consulting Group's *Innovation 2006*.
2. Membership in the Project Management Institute stood at 485,000 with 762,000 credentialed PMPs as of March 2017 and has grown from 237,000 in 2007. See the Project Management Institute's Member Fact Sheet.
3. This is a short discussion of change in modern business. For a longer and more strategic treatment, See Brown and Eisenhardt's *Competing on the Edge*.
4. Arthur Fry and Spencer Silver invented the sticky note in 1970, based on Silver's 1968 invention of the "low-tack" adhesive involved. It was 1979 before 3M found a means for marketing it successfully. See en.wikipedia.org/wiki/Spencer_Silver and inventors.about.com/library/inventors/blpostit.htm (both accessed September 24, 2018).
5. See Smith and Reinertsen's *Developing Products in Half the Time*.
6. For a contemporary treatment, see Cooper, Edgett, and Kleinschmidt's *Portfolio Management for New Products*. For good coverage of the balance between new and mature products, see Roussel, Saad, and Erickson's *Third Generation R&D*.
7. See Boehm's *Software Engineering Economics*, p. 40.
8. See Boehm and Turner's *Balancing Agility and Discipline*, pp. 218–219.
9. See Terwiesch, Loch, and De Meyer's "Exchanging Preliminary Information in Concurrent Engineering." They provide detailed cost-of-change information for three automotive components, and their results support the 10x/phase rule. However, this is an example of the worst part of the Pareto principle. They only investigate the two most expensive transitions in the development process (toolmaking), which together yield a 100x cost increase.

10. See Larman's *Agile and Iterative Development*.
11. See agilemanifesto.org/principles.html, accessed September 24, 2018.
12. This discussion draws from Beck's *Extreme Programming Explained*. Although this book comes in two editions (dated 2000 and 2005), it is actually more like parts 1 and 2. The first edition lays out the basics, and the second edition, requiring some existing knowledge of XP, describes it more broadly, provides case studies, and covers more advanced topics, such as the roles of project planning and software testing and the scalability of XP to large teams, dispersed teams, and life-critical applications. Consequently, this discussion follows the first edition more than the second one. Also see en.wikipedia.org/wiki/Extreme_Programming (accessed September 24, 2018).
13. Although commonly used, the term *pair programming* is misleading even for software development, because these pairs not only program but design, test, integrate, and document. Compare to the shift from *concurrent engineering* to *concurrent development*. Thus I use simply *pairing* from here on.
14. See Boehm and Turner's *Balancing Agility and Discipline*, pp. 230–232.
15. See Beck's *Extreme Programming Explained* (First Edition), pp. 64–70, for more such relationships..
16. en.wikipedia.org/wiki/Extreme_Programming (Origins section) or informit.com/articles/article.aspx?p=20972&rl=1 (both accessed September 24, 2018).
17. See Beck's *Extreme Programming Explained* (First Edition), pp. 29–34. The second edition adds respect and suggests that others, such as safety, predictability, and quality of life, may be important in certain situations.
18. See Boehm and Turner's *Balancing Agility and Discipline*, p. 231.
19. See Eckstein's *Agile Software Development in the Large*.

Chapter Two

1. I use *specification* and *requirement* interchangeably to mean the written description of the product going into development, but I know that some people make a clear distinction here. Some say that requirements are the wish list at the beginning of the project, while specifications appear in the catalog after commercialization. Others believe that requirements are for your product, and specifications relate to purchased components or materials. Please picture the term you prefer for the up-front description of what the product should be.
2. Basic data are from Thomke and Reinertsen's "Agile Product Development," updated via personal communication with Donald Reinertsen, December 11, 2006.
3. See MacCormack's "Developing Products on 'Internet Time': The Anatomy of a Flexible Development Process," "How Internet Companies Build Software," and "Creating a Fast and Flexible Process: Empirical Research Suggests Keys to Success."

4. See Boehm's "Prototyping Versus Specifying: A Multiproject Experiment" for details.
5. Source: The Standish Group, standishgroup.com, as reported at the 3rd International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2002), Alghero, Sardinia, Italy, May 2002.
6. Lynn and Reilly's *Blockbusters* lists a clear and stable product vision as one of five factors that distinguished what they call "blockbuster" products from others in a ten-year study of seven hundred development projects. For more examples of vision statements and how to create them, see their book.
7. From Jeffrey K. Liker, *The Toyota Way*, McGraw-Hill, 2004 (Figure 5-4). Reproduced with permission of The McGraw-Hill Companies.
8. See Morgan and Liker's *Toyota Product Development System*, p. 123.
9. See Morgan and Liker's *Toyota Product Development System*, p. 260. Also, private communication with the authors, November 6, 2006.
10. See Sobek's "Principles That Shape Product Development Systems," p. 226.
11. Cooper's *The Inmates Are Running the Asylum* describes this technique.
12. For more on this topic, see Cockburn's *Writing Effective Use Cases*.
13. For more on this topic, see Cohn's *User Stories Applied*.
14. You can often drive user stories to a higher, more change-resistant level by writing "epics," as Cohn describes.
15. See Hohmann's *Innovation Games*.
16. *Customers* (those who pay for your products) are not necessarily *users* (those who use them). For consumer products, the two are often the same, but for industrial products, they are usually different: a buyer or a manager may purchase the computer on your desk at work, but you have to use it and cope with it every day. I use the terms interchangeably, but please apply the pertinent one.
17. Sometimes Marketing and Sales personnel are nervous about letting engineers talk to customers, fearing that they might say something inappropriate. If so, train those involved using Hohmann's guidance described in *Beyond Software Architecture*, p. 60.
18. See Scoble and Israel's *Naked Conversations* for much more on business blogging.
19. In Hohmann's *Innovation Games*.
20. Both Toyota examples are from Morgan and Liker's *Toyota Product Development System*, p. 30.
21. For example, see Mariampolski's *Ethnography for Marketers*, and Perry and her colleagues' "Creating the Customer Connection."
22. For more on lead users, see von Hippel's *Sources of Innovation*, and for other examples and the 3M infection study, see his "Creating Breakthroughs at 3M."
23. See Christensen's *The Innovator's Dilemma*.
24. See Highsmith's *Agile Project Management*, p. 13.

Chapter Three

1. From Stevens, Myers, and Constantine, “Structured Design,” p. 117. Quotation attributed to Constantine, who is considered the originator of the structured design technique used in software development and was the teacher of Stevens and Myers.
2. Adapted from The PDMA Glossary for New Product Development. See pdma.org/page/glossary_access2#P (accessed September 24, 2018).
3. For more on platforms and platform architecture, see Meyer and Lehnerd’s *Power of Product Platforms* or Feitzinger and Lee’s “Mass Customization at Hewlett-Packard: The Power of Postponement.”
4. This shift between integral and modular architectures may appear to contradict Christensen and Raynor’s observations in *The Innovator’s Solution* (pp. 127–137). These authors work at the level of industries and describe how, in the early stages of an innovation (when I suggest that the need for flexibility and thus modularity is greatest), the product’s performance falls below customer desires, so designers squeeze performance from the design. (integral architecture). Later, as the technology improves, performance exceeds needs and customers are unwilling to pay extra for performance, so the industry shifts to a modular architecture that allows more flexibility in tuning the product to individual needs at minimal cost. Observe that this strategic modularity aims more at changes during manufacture and distribution, that is, it is more of a platform architecture approach. In the early stages of an innovation, there is still a great need for designers to isolate areas of uncertainty and provide for reserve performance where needs are likely to grow, so wise designers apply modularity selectively at the design level, especially in the early stages of an innovation. This illustrates why flexibility techniques must be applied selectively so that performance will not suffer excessively just when Christensen and Raynor advise that it is most important.
5. Quotation from ferrariusa.com/design_f430text.php, accessed April 23, 2007.
6. These techniques are old. David Parnas provided an elegant (but technical) description of them for software nearly three decades ago. See Parnas’ “Designing Software for Ease of Extension and Contraction.”
7. Here I follow Ulrich and Eppinger’s *Product Design and Development*.
8. For more on considering interfaces as design rules, see Baldwin and Clark’s *Design Rules*.
9. A similar example is the naming of Windows XP covered earlier in this chapter.
10. See Thomke’s “Role of Flexibility in the Development of New Products” for further discussion.

Chapter Four

1. Sources, respectively: Thomke's *Experimentation Matters*, p. 6; nmlites.org/standards/science/glossary_2.htm; and www.math.tamu.edu/FiniteMath/FinalBuild/Fall2001/Module9/Introduction0.html.
2. Source: Archibald Lemon Cochrane. See his *Effectiveness and Efficiency*, p. 43.
3. See Thomke's *Experimentation Matters*, pp. 211–214.
4. For an extensive discussion of exploration and hypothesis-based experimentation, see Garvin's *Learning in Action*, chapter 5 (specifically what he calls the probe-and-learn process).
5. See Iansiti and MacCormack's "Developing Products on Internet Time."
6. The Orion case study was published by Z Corporation in 2001 but is no longer available.
7. See Thomke's "Capturing the Real Value of Innovation Tools."
8. See Smith and Reinertsen's *Developing Products in Half the Time*, chapter 2.
9. See, for example, Montgomery's *Design and Analysis of Experiments*.
10. The lock analogy was inspired by Thomke's *Experimentation Matters*, p. 110. For more on parallel versus sequential strategies, see Thomke and Bell's "Sequential Testing in Product Development."
11. *Testing* in its broadest sense includes testing an idea, a prototype, or even a hunch. Here I consider the narrower interpretation of testing a design, a product, or a part of a product.
12. See Peters and Austin's *Passion for Excellence*, p. 130. My second engineering job was for a manufacturer of aircraft engines twenty-plus years before their book appeared. At that time the test used a two-pound seagull, but the procedure was the same. This was regarded as the final test of an engine.

Chapter Five

1. Yogi Berra was a famous American baseball player from several decades ago, but today he is known better for his illogical but direct manner of speaking.
2. Engineering changes do *not* constitute flexibility. Engineering changes originate internal to engineering and in most cases arise from poor engineering judgment. Flexibility stems from changes external to engineering. Thus flexibility is beneficial when external change is likely, but engineering changes are usually an indication of mistakes.
3. See Ward, Liker, Cristiano, and Sobek's "Second Toyota Paradox." This article is also the source of the concept for Figure 5.4.
4. See Pugh's *Total Design*, section 4.8.

7. For additional details, see Morgan and Liker's *Toyota Product Development System*, pp. 269–274. For examples of A3 reports, guidance on writing them, and templates for writing the basic types, search the Web for “A3 report.”
8. See Sobek, Ward, and Liker's “Toyota's Principles of Set-Based Concurrent Engineering.”
9. The examples in this and the following few paragraphs come from Durward Sobek's Ph.D. dissertation.
10. Suggested by Katherine Radeka of Whittier Consulting Group.
11. See Ward, Liker, Cristiano, and Sobek's “Second Toyota Paradox.”

Chapter Six

1. From Turner and Boehm's “People Factors in Software Management.”
2. The book is Boehm and Turner's *Balancing Agility and Discipline*.
3. See Boehm's *Software Engineering Economics*.
4. See Quinn's *Building the Bridge As You Walk On It*. However, only the title of this book is pertinent here, as the book deals with personal change rather than project change.
5. See MacCormack's “How Internet Companies Build Software” for more on experience.
6. See Loch, DeMeyer, and Pich's *Managing the Unknown*, p. 41.
7. See Cockburn's *Agile Software Development*, pp. 14–18, and Boehm and Turner's *Balancing Agility and Discipline*, p. 48.
8. This has no connection with Douglas McGregor's Theory X, which postulates managers who regard workers as lazy and unwilling to work without strong structure and control.
9. This figure results from a multi-year study in a major electronics company that categorized engineers' activities as either adding or not adding value to their development projects.
10. In a 2006 article of the same name, Scott Ambler uses the term *generalizing specialists* and discusses both their benefits to the team and the techniques for cultivating them.
11. See agilemanifesto.org/principles.html, accessed September 24, 2018.
12. See Figure 8-7 (p. 156) of Smith and Reinertsen's *Developing Products in Half the Time*.
13. Many people call this a “virtual team,” but I dislike this term. Teams exist to enhance performance, and it clouds the performance issue to use a term that means “being such in essence or effect though not formally recognized or admitted” (webster.com, accessed September 24, 2018). Are the members virtual, are the activities virtual, or what is it that exists only in essence or effect?

7. See Allen's *Managing the Flow of Technology*. His more contemporary *Organization and Architecture of Innovation* (pp. 58–61) reinforces his earlier work and concludes that electronic communication media, such as e-mail, lack the fidelity to substitute for face-to-face communication as distance increases.
8. As suggested at the beginning of this section, because effectiveness drops off gradually, various distances are used to define co-location. Reinertsen is somewhat more liberal in his definition than I am. Stephanie Teasley and her colleagues (quoted a few paragraphs later) are still more liberal.
9. The Olsons' article appeared in *Human-Computer Interaction* in 2000.
10. This work was published by Teasley, Covi, Krishnan, and Olson as "How Does Radical Collocation Help a Team Succeed?" Both of the quotations that follow this paragraph are reprinted with permission from that article, copyright ACM.
11. See Cockburn's *Agile Software Development*, pp. 84–88.
12. See Williams and Kessler's *Pair Programming Illuminated*.
13. From Duarte and Snyder's *Mastering Virtual Teams*. Used with permission.
14. See Duarte and Snyder's *Mastering Virtual Teams*, p. 42.

Chapter Seven

1. Quoted in Maier and Reichtin's *Art of Systems Architecting*, p. 272. Robert Spinrad is a retired vice president of technology strategy and director of the legendary PARC laboratories at Xerox. He is also a member of the prestigious National Academy of Engineering in the United States.
2. For example, see Deck's "Decision Making: The Overlooked Competency in Product Development."
3. "The last responsible moment" was coined in about 2000 by the Lean Construction Institute in work they were doing with the British Airports Authority to create the construction process for Terminal 5 at London's Heathrow Airport. They needed flexibility because the airlines' strategic plans were likely to change numerous times during the eight years needed to build the terminal. In *Lean Software Development*, Poppendieck and Poppendieck brought the term into agile software development, and I elaborate further.
4. In developing this section, I consulted some experts on building consensus. Each had a somewhat different way of using the consensus gradient. Consequently, what you see here is truly a consensus version of the topic, keeping the elements separate so that you can assemble them to suit your needs.
5. For this section, I am indebted to chapter 6 of Savage's excellent book, *Decision Making with Insight*. This chapter introduces decision trees and provides XLTree, an Excel add-in, for creating them. All decision trees in this chapter were created using XLTree. Unfortunately, XLTree isn't compatible with contemporary operating systems (Windows 10, for example), but plenty of other such tools are available.

7. For an introduction to utility theory and utility measures, see Savage's *Decision Making with Insight*, pp. 194–195.
8. For an example of real options applied to the product development process, see Huchzermeier and Loch's "Project Management Under Risk."
9. For a starter on real options applied to new product development, see Faulkner's "Applying 'Options Thinking' to R&D Valuation," Angelis' "Capturing the Option Value of R&D," and van Putten's "Making Real Options Really Work." Several other helpful articles are available in *Research-Technology Management* and the *Harvard Business Review*.
10. Traditionally, such valuations are made using discounted cash flow (DCF) methods, which assume the project will be completed according to the original plan. Even when used with a phased development process, where the purpose of the phases is that one can kill the project rather than continuing to invest, the project is still evaluated on the assumption of completing the plan. In contrast, the real options approach allows a project to be evaluated more flexibly by making future investments contingent on interim results.
11. For a discussion of how real options are equivalent to decision trees, see Faulkner's "Applying 'Options Thinking' to R&D Valuation."

Chapter Eight

1. The referenced guide is the Project Management Institute's *Guide to the Project Management Body of Knowledge*.
2. Several other books on agile project management have appeared in the last few years. I list DeCarlo's and Highsmith's because they generalize beyond software development better than most of the others, I believe. Interestingly, an *Agile Practice Guide*, jointly published by PMI and the Agile Alliance, appeared in 2017, apparently because agile had become such an important topic in PMI that it could no longer be ignored. This guide covers a few agile practices, such as stand-up meetings (Chapter Six in this book) and retrospectives (Chapter Eight), but it doesn't mention core topics, such as customer involvement (Chapter Two), experimentation (Chapter Four), or decision making (Chapter Seven).
3. Chapter Nine shows how to combine structured and flexible development processes in the same project, depending on the project's specific demands. You can use the same approach to adjust project management to the specific characteristics of a project.
4. See the Project Management Institute's *Guide to the Project Management Body of Knowledge*, p. 356.
5. See Portny's *Project Management for Dummies* (Second Edition), p. 14.
6. See the Project Management Institute's *Guide to the Project Management Body of Knowledge*, p. 369.
7. Chapter Nine covers anticipation relative to adaptation, which is investing in the capability to react quickly when anticipation is not possible or cost-effective.

8. See Figure 9.4 for an illustration of the way a project naturally shifts from anticipation to planning as it progresses.
9. For additional information on this method, see Githens's "Using a Rolling Wave for Fast and Flexible Development."
10. For more detail on agile loose-tight planning, see Cohn's *Agile Estimating and Planning*.
11. See Thomke's *Experimentation Matters*, pp. 168–169.
12. See Highsmith's *Agile Project Management*, p. 42.
13. Smith and Merritt's *Proactive Risk Management* is a good example of this literature.
14. Although most agilists have replaced waterfall (phased) processes with iterative ones, Jim Highsmith has created a means of combining a phased governance process (for managing project investments) with iterative development (for flexibility). See his "Agile for the Enterprise: From Agile Teams to Agile Organizations."
15. See Loch, DeMeyer, and Pich's *Managing the Unknown*. Although valuable, this book is unfortunately also expensive. However, you can read a lengthy review of it gratis at <https://www.strategy2market.com/Preston-Smith/Book-Reviews/Managing-the-Unknown/> (accessed September 25, 2018).
16. See Weick and Sutcliffe's *Managing the Unexpected*.
17. Summarized with permission from Weick and Sutcliffe's *Managing the Unexpected*, pp. 10–17.
18. This metric was inspired by Iansiti's "Shooting the Rapids," Figure 1.
19. I intentionally use *feel* here, but I recognize that some people are quite uneasy in expressing their feelings or hearing about others' feelings. In this case, please see Derby and Larsen's *Agile Retrospectives* (p. 10) for ways of obtaining this information without using that *f* word directly.
20. See Morgan and Liker's *Toyota Product Development System*, p. 211.

Chapter Nine

1. See Boehm and Turner's *Balancing Agility and Discipline*, pp. 36–37 and p. 152.
2. See note 14 in Chapter Eight.
3. See Cockburn's "Learning from Agile Software Development."
4. For a description of how IT might be used to capture and manage product development knowledge, see McGrath's *Next Generation Product Development*.
5. For more on how Toyota manages tacit knowledge, see Morgan and Liker's *Toyota Product Development System*, pp. 204–205, 229, and 279–280. I describe A3 reports and engineering checklists in Chapter Five.
6. As discussed in the Introduction, using *structured* as the opposite of *flexible* has difficulties in that flexible development is very structured in certain

subtle ways. Agilists speak of high- and low-ceremony processes. Nevertheless, *traditional* (used in the rest of the book) seems to miss the mark when discussing process, so I will proceed with *structured* here, knowing that it is not a perfect antonym.

7. See Boehm and Turner's *Balancing Agility and Discipline*, chapter 5. For non-software projects, their approach must be modified in two ways. One is to shift it to a physical product, and the other is to narrow it to a focus on flexibility (Boehm and Turner consider other differences between agile and traditional approaches, such as scalability). Also, I do not follow their complete five-step process. For an alternative means of adjusting the process used to the needs of a specific project, also see the Project Analyzer discussed at the end of Chapter One.
8. The infrared measurement technologies and algorithms described here are fictitious.
9. Goldratt and Cox's *The Goal* (a business novel) is the classic on this topic, but more descriptive material can be found on the Web, for instance, en.wikipedia.org/wiki/Theory_of_Constraints (accessed September 25, 2018).
10. For bulk arrivals, there is no formula for the answer, so I used the Extend discrete-event simulation package from Imagine That, Inc. and covered in Savage's *Decision Making with Insight*. I assumed that arrivals appear in uniformly distributed clumps of one to five items (mean of three), as well as being exponentially distributed in time. Extend generated Figure 9.7 too.
11. From "What Testers Can Do About Technical Debt" by Johanna Rothman, <https://www.cmcrossroads.com/article/what-testers-can-do-about-technical-debt-part-1> and <https://www.cmcrossroads.com/article/what-testers-can-do-about-technical-debt-part-2>, accessed September 25, 2018.

Chapter Ten

1. Brooks managed development of the OS/360 software system. The quote comes from his *Mythical Man-Month*, p. 242. "System" to him means software or hardware.
2. From Kotter and Cohen's *Heart of Change*, p. 40.
3. For more on pilot projects, see Smith and Reinertsen's *Developing Products in Half the Time*, chapter 15.
4. From Bridges' *Managing Transitions*, p. 37.
5. From Kotter and Cohen's *Heart of Change*, p. 2.
6. From Patterson's *Leading Product Innovation*, pp. 262–264.
7. See Schaffer's *Breakthrough Strategy* for more on this momentum-building process.
8. From Morgan and Liker's *Toyota Product Development System*, p. 227.
9. For more on patterns, see en.wikipedia.org/wiki/Software_pattern, accessed September 25, 2018.
10. From Manns and Rising's *Fearless Change*, p. 5.

Chapter Eleven

1. This method of managing changes is strongly connected with the technique used to manage risk in a project. In fact, changes are just a subset of the risks in a project. A strong body of knowledge is available in the field of project risk management field. I have written a book on this subject that parallels the technique described in this chapter, and The Project Management Institute has published many books about project risk management, including their *PMBOK Guide*. If you search the Internet for help on the subject, be sure to search for *project risk management*; *risk management* alone will bring you information mostly about the insurance industry. But please remember that none of this substitutes for building an environment that accommodates change.
2. Note 1 mentions that managing anticipated changes is quite like project risk management. Those managing project risks may not have a template like Figure 11.1, but they follow the same steps. The big difference is that they do not have a Benefits box. Although they sometimes mention benefits (called opportunities in project risk management), very seldom do benefits actually occur. Anticipated changes, in contrast, can have substantial benefits, sometimes outweighing the costs.
3. Although this book is written in U.S. English, it is international in scope. Thus, I use a variety of currencies. U.S. dollars, euros, and Japanese yen have already been used, so the examples in this chapter employ the Chinese yuan or RMB (which translates into “people’s money” in Chinese).
4. This value was obtained by using the cost of delay calculation described in chapter 2 of Smith and Reinertsen.
5. See chapter 2 of Smith and Reinertsen for a detailed explanation.
6. Although this may look complicated, it is actually a relatively simple model. It allows for only one benefit and one cost item per anticipated change. If you have more than one benefit or multiple costs associated with a single change, you can handle this by placing an i subscript on pertinent quantities below and summing over i , where i runs from 1 to the number of benefits or costs involved.
7. Chosen because it is the only change of the three to have both a benefit and a cost.
8. There are often multiple benefits or costs for a given change. Usually, some simple analysis will show that one benefit and one cost are dominant, and you can proceed with them alone. If one doesn’t stand out, see note 6 above.
9. See the section entitled Tacit Knowledge in Chapter Nine.
10. Readers outside North America may need an explanation. Canada has a national health care system, which the United States lacks, so drug prices are regulated at much lower levels (for the same drugs) in Canada. Consequently, Canadian pharmacies have a huge market serving U.S. customers online. But drug companies and the regulatory authority (the Food and Drug Administration) in the United States oppose this, as it undercuts their power and authority. This creates volatile conditions as the various opposing parties act constantly to enhance their positions.

Bibliography

- Allen, Thomas J. *Managing the Flow of Technology*. Cambridge, MA: MIT Press, 1977.
- Allen, Thomas J., and Gunter W. Herr. *The Organization and Architecture of Innovation: Managing the Flow of Technology*. Amsterdam: Butterworth-Heinemann, 2007.
- Ambler, Scott W. "Generalizing Specialists: Improving Your IT Career Skills," 2006. <http://agilemodeling.com/essays/generalizingSpecialists.htm>.
- Angelis, Diana I. "Capturing the Option Value of R&D." *Research-Technology Management* 43(4): 31–34 (July–August 2000).
- Avery, Christopher M. *Teamwork Is an Individual Skill: Getting Your Work Done When Sharing Responsibility*. San Francisco: Berrett-Koehler, 2001.
- Baldwin, Carliss Y., and Kim B. Clark. *Design Rules: The Power of Modularity*. Cambridge, MA: MIT Press, 2000.
- Beck, Kent. *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley, 2000.
- Beck, Kent, and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. (Second Edition). Boston: Addison-Wesley, 2005.
- Boehm, Barry W. *Software Engineering Economics*. Upper Saddle River, NJ: Prentice Hall, 1981.
- Boehm, Barry W., Terance E. Gray, and Thomas Seewaldt. "Prototyping Versus Specifying: A Multiproject Experiment." *IEEE Transactions on Software Engineering* SE-10(3): 290–302 (May 1984).
- Boehm, Barry W., Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford K. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani, and Chris Abts. *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall, 2000.
- Boehm, Barry, and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley, 2004.
- Boston Consulting Group. *Innovation 2006*. Boston: Boston Consulting Group, Inc., 2006.

- Bridges, William, *Managing Transitions: Making the Most of Change* (Second Edition). Cambridge, MA: Da Capo Press, 2003.
- Brooks, Frederick P., Jr., *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Reading, MA: Addison-Wesley, 1995.
- Brown, Shona L, and Kathleen M. Eisenhardt. *Competing on the Edge: Strategy as Structured Chaos*. Boston: Harvard Business School Press, 1998.
- Christensen, Clayton M. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Boston: Harvard Business School Press, 1997.
- Christensen, Clayton M., and Michael E. Raynor. *The Innovator's Solution: Creating and Sustaining Successful Growth*. Boston: Harvard Business School Press, 2003.
- Cochrane, Archibald Lemon. *Effectiveness and Efficiency: Random Reflections on Health Services*. London: Nuffield Provincial Hospitals Trust, 1972.
- Cockburn, Alistair. *Writing Effective Use Cases*. Boston: Addison-Wesley, 2000.
- Cockburn, Alistair. *Agile Software Development*. Boston: Addison-Wesley, 2002.
- Cockburn, Alistair. "Learning from Agile Software Development—Part One." *Crosstalk: The Journal of Defense Software Engineering* 15(10): 10–14 (October 2002).
- Cohn, Mike. *User Stories Applied: For Agile Software Development*. Boston: Addison-Wesley, 2004.
- Cohn, Mike. *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall, 2006.
- Cooper, Alan. *The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*. Indianapolis: SAMS, 1999.
- Cooper, Robert G. "Your NPD Portfolio May Be Harmful to Your Business's Health," *Visions* 29(2): 22–26 (April 2005).
- Cooper, Robert G., Scott J. Edgett, and Elko J. Kleinschmidt. *Portfolio Management for New Products*, (Second Edition). Cambridge, MA: Perseus Books, 2001.
- DeCarlo, Doug. *eXtreme Project Management: Using Leadership, Principles, and Tools to Deliver Value in the Face of Reality*. San Francisco: Jossey-Bass, 2004.
- Deck, Mark J. "Decision Making: The Overlooked Competency in Product Development." In *The PDMA ToolBook 1 for New Product Development*, Paul Belliveau, Abbie Griffin, and Stephen Somermeyer, eds. New York: Wiley, 2002, pp. 165–185.
- Derby, Esther, and Diana Larsen. *Agile Retrospectives: Making Good Teams Great*. Raleigh, NC: Pragmatic Bookshelf, 2006.
- Duarte, Deborah L., and Nancy Tennant Snyder. *Mastering Virtual Teams: Strategies, Tools, and Techniques That Succeed* (Third Edition). San Francisco: Jossey-Bass, 2006.

- Eckstein, Jutta. *Agile Software Development in the Large: Diving into the Deep*. New York: Dorset House, 2004.
- Faulkner, Terrence W. "Applying 'Options Thinking' to R&D Valuation." *Research-Technology Management* 39(3): 50–56 (May–June 1996).
- Feitzinger, Edward, and Hau L. Lee. "Mass Customization at Hewlett-Packard: The Power of Postponement." *Harvard Business Review* 75(1): 116–121 (January–February 1997).
- Garvin, David A. *Learning in Action: A Guide to Putting the Learning Organization to Work*. Boston: Harvard Business School Press, 2000.
- Githens, Gregory D. "Using a Rolling Wave for Fast and Flexible Development." In *The PDMA ToolBook 3 for New Product Development*, Abbie Griffin and Stephen Somermeyer, eds. 397–415. Hoboken, NJ: Wiley, 2007.
- Goldratt, Eliyahu M., and Jeff Cox, *The Goal* (Third Edition). Great Barrington, MA: North River Press, 2004.
- Highsmith, Jim. *Agile Project Management: Creating Innovative Products*. Boston: Addison-Wesley, 2004.
- Highsmith, Jim. "Agile for the Enterprise: From Agile Teams to Agile Organizations." *Cutter Consortium Agile Project Management Advisory Service Executive Report* 6(1) (2005).
- Hohmann, Luke. *Beyond Software Architecture: Creating and Sustaining Winning Solutions*. Boston: Addison-Wesley, 2003.
- Hohmann, Luke. *Innovation Games: Creating Breakthrough Products Through Collaborative Play*. Upper Saddle River, NJ: Addison-Wesley, 2007.
- Huchzermeier, Arnd, and Christoph H. Loch. "Project Management Under Risk: Using the Real Options Approach to Evaluate Flexibility in R&D." *Management Science* 47(1): 85–101 (January 2001).
- Iansiti, Marco. "Shooting the Rapids: Managing Product Development in Turbulent Environments." *California Management Review* 38(1): 37–58 (Fall 1995).
- Iansiti, Marco, and Alan MacCormack. "Developing Products on Internet Time." *Harvard Business Review* 75(5): 108–117 (September–October 1997).
- Kerth, Norman L. *Project Retrospectives: A Handbook for Team Reviews*. New York: Dorset House, 2001.
- Kotter, John P., and Dan S. Cohen. *The Heart of Change: Real-Life Stories of How People Change Their Organizations*. Boston: Harvard Business School Press, 2002.
- Larman, Craig. *Agile and Iterative Development: A Manager's Guide*. Boston: Addison-Wesley, 2004.
- Loch, Christoph, Arnoud DeMeyer, and Michael T. Pich. *Managing the Unknown: A New Approach to Managing High Uncertainty and Risk in Projects*. Hoboken, NJ: Wiley, 2006.
- Lynn, Gary S., and Richard R. Reilly. *Blockbusters: The Five Keys to Developing Great New Products*. New York: Harper Business, 2002.

- MacCormack, Alan. "How Internet Companies Build Software." *Sloan Management Review* 42(2): 75–84 (Winter 2001).
- MacCormack, Alan. "Creating a Fast and Flexible Process: Empirical Research Suggests Keys to Success." *Product Development Best Practices Report* 10(8): 1–4 (August 2003).
- MacCormack, Alan, Roberto Verganti, and Marco Iansiti. "Developing Products on 'Internet Time': The Anatomy of a Flexible Development Process." *Management Science* 47(1): 133–150 (January 2001).
- Maier, Mark W., and Eberhardt Rechtin. *The Art of Systems Architecting* (Second Edition). Boca Raton, FL: CRC Press, 2002.
- Manns, Mary Lynn, and Linda Rising. *Fearless Change: Patterns for Introducing New Ideas*. Boston: Addison-Wesley, 2005.
- Mariampolski, Hy. *Ethnography for Marketers: A Guide to Consumer Immersion*. Thousand Oaks, CA: Sage, 2006.
- McGrath, Michael E. *Next Generation Product Development: How to Increase Productivity, Cut Costs, and Reduce Cycle Times*. New York: McGraw-Hill, 2004.
- Meyer, Marc H., and Alvin P. Lehnerd. *The Power of Product Platforms: Building Value and Cost Leadership*. New York: Free Press, 1997.
- Montgomery, Douglas C. *Design and Analysis of Experiments*. Hoboken, NJ: Wiley, 2005.
- Morgan, James M., and Jeffrey K. Liker. *The Toyota Product Development System: Integrating People, Process, and Technology*. New York: Productivity Press, 2006.
- Olson, Gary M., and Judith S. Olson. "Distance Matters." *Human-Computer Interaction* 15(2–3): 139–178 (2000).
- Parnas, David L. "Designing Software for Ease of Extension and Contraction." *IEEE Transactions on Software Engineering* SE-5(2): 128–138 (March 1979).
- Patterson, Marvin L. *Leading Product Innovation: Accelerating Growth in a Product-Based Business*. New York: Wiley, 1999.
- Perry, Barbara, Cara L. Woodland, and Christopher W. Miller. "Creating the Customer Connection: Anthropological/Ethnographic Needs Discovery." In *The PDMA ToolBook 2 for New Product Development*, Paul Belliveau, Abbie Griffin, and Stephen M. Somermeyer, eds. 201–234. Hoboken, NJ: Wiley, 2004.
- Peters, Tom, and Nancy Austin. *A Passion for Excellence*. New York: Random House, 1985.
- Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Boston: Addison-Wesley, 2003.
- Portny, Stanley E. *Project Management for Dummies* (Second Edition). Indianapolis, IN: Wiley, 2007.
- Project Management Institute. *A Guide to the Project Management Body of Knowledge* (Third Edition). Newtown Square, PA: Project Management Institute, 2004.

- Project Management Institute. *Agile Practice Guide*. Newtown Square, PA: Project Management Institute, 2017.
- Project Management Institute. Member Fact Sheet. Available online (search on “PMI Fact File”). Accessed September 26, 2018.
- Pugh, Stuart. *Total Design*. Wokingham, UK: Addison-Wesley, 1991.
- Quinn, Robert E. *Building the Bridge As You Walk On It: A Guide for Leading Change*. San Francisco: Jossey-Bass, 2004.
- Reinertsen, Donald G. *Managing the Design Factory: A Product Developer's Toolkit*. New York: Free Press, 1997.
- Roussel, Philip A., Kamal N. Saad, and Tamara J. Erickson. *Third Generation R&D: Managing the Link to Corporate Strategy*. Boston: Harvard Business School Press, 1991.
- Savage, Sam L. *Decision Making with Insight*. Belmont, CA: Brooks/Cole, 2003.
- Schaffer, Robert H. *The Breakthrough Strategy: Using Short-Term Success to Build the High-Performance Organization*. Cambridge, MA: Ballinger, 1988.
- Scoble, Robert, and Shel Israel. *Naked Conversations: How Blogs Are Changing the Way Businesses Talk with Customers*. Hoboken, NJ: Wiley, 2006.
- Smith, Preston G., and Guy M. Merritt. *Proactive Risk Management: Controlling Uncertainty in Product Development*. New York: Productivity Press, 2002.
- Smith, Preston G., and Donald G. Reinertsen. *Developing Products in Half the Time* (Second Edition). New York: Wiley, 1998.
- Sobek, II, Durward Kenneth. “Principles That Shape Product Development Systems: A Toyota-Chrysler Comparison.” Ph.D. dissertation, University of Michigan, 1997.
- Sobek, II, Durward K., Allen C. Ward, and Jeffrey K. Liker. “Toyota’s Principles of Set-Based Concurrent Engineering.” *Sloan Management Review* 40(2): 67–83 (Winter 1999).
- Stevens, Wayne G., Glen J. Myers, and Larry L. Constantine. “Structured Design,” *IBM Systems Journal* 13(2): 115–139 (May 1974).
- Teasley, Stephanie, Lisa Covi, M. S. Krishnan, and Judith S. Olson. “How Does Radical Collocation Help a Team Succeed?” Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, pp. 339–346, 2000.
- Terwiesch, Christian, Christoph H. Loch, and Arnoud De Meyer. “Exchanging Preliminary Information in Concurrent Engineering: Alternative Coordination Strategies.” *Organization Science* 13(4): 402–419 (July/August 2002).
- Thomke, Stefan H. “The Role of Flexibility in the Development of New Products: An Empirical Study.” *Research Policy* 26(1): 105–119 (March 1997).
- Thomke, Stefan H. *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation*. Boston: Harvard Business School Press, 2003.
- Thomke, Stefan H. “Capturing the Real Value of Innovation Tools.” *Sloan Management Review* 47(2): 24–32 (Winter 2006).

- Thomke, Stefan, and David E. Bell. "Sequential Testing in Product Development." *Management Science* 47(2): 308–323 (February 2001).
- Thomke, Stefan, and Donald Reinertsen. "Agile Product Development: Managing Development Flexibility in Uncertain Environments." *California Management Review* 41(1): 8–30 (Fall 1998).
- Turner, Richard, and Barry Boehm. "People Factors in Software Management: Lessons from Comparing Agile and Plan-Driven Methods." *Crosstalk: The Journal of Defense Software Engineering* 16(12): 4–8 (December 2003).
- Ulrich, Karl T., and Steven D. Eppinger. *Product Design and Development* (Third Edition). New York: McGraw-Hill, 2003.
- van Putten, Alexander B., and Ian C. Macmillan. "Making Real Options Really Work." *Harvard Business Review* 82(12): 134–141 (December 2004).
- von Hippel, Eric. *The Sources of Innovation*. New York: Oxford University Press, 1988.
- von Hippel, Eric, Stefan Thomke, and Mary Sonnack. "Creating Breakthroughs at 3M." *Harvard Business Review* 77(5): 47–57 (September–October 1999).
- Ward, Allen, Jeffrey K. Liker, John J. Cristiano, and Durward K. Sobek II. "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster." *Sloan Management Review* 36(3): 43–61 (Spring 1995).
- Weick, Karl E., and Kathleen M. Sutcliffe. *Managing the Unexpected*. San Francisco: Jossey-Bass, 2001.
- Wheelwright, Steven C., and Kim B. Clark. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*. New York: Free Press, 1992.
- Williams, Laurie, and Robert Kessler. *Pair Programming Illuminated*. Boston: Addison-Wesley, 2003.

Customer Council

Many people have contributed generously to this book, for which I am most grateful. Without them, the book would not have been possible. However, I would like to acknowledge in particular a “customer council” of lead users of flexibility techniques, the people who have played a key role in reviewing chapters, suggesting improvements, offering examples, and providing sources of associated material. I am especially grateful to these individuals on the customer council, each of whom has made an important contribution to the book:

| <i>Name</i> | <i>Affiliation</i> | <i>Country</i> |
|-------------------|---|----------------|
| Bob Becker | Product Development Advantage Group | United States |
| Chuck Blevins | LifeScan, Inc. (Johnson & Johnson) | United States |
| Jim Callahan, PMP | C-Cor, Incorporated | United States |
| Alan Chachich | Breakthrough NPD | United States |
| Frankie Chan | T&K Industrial Company Limited (China) | Hong Kong |
| Mike Clem | Cordis Accelerated Medical Ventures (Johnson & Johnson) | United States |
| Mike Cohn | Mountain Goat Software | United States |
| Mike Dowson | Draeger Safety | United Kingdom |
| Ricco Estanislao | Johnson & Johnson Worldwide Emerging Markets Innovation Center | China |

| | | |
|----------------------------------|---|----------------|
| John Farnbach | Silver Streak Partners LLC | United States |
| Gregory D. Githens, PMP, NPDP | Catalyst Management Consulting, LLC | United States |
| Mike Griffiths | Independent Consultant | Canada |
| David Gunderson | Micro Power Electronics, Inc. | United States |
| Greg Krisher | GE Water & Process Technologies, Analytical Instruments | United States |
| Jeff Oltmann, PMP | Synergy Professional Services, LLC | United States |
| Barry Papoff | Harris Corporation | Canada |
| David Petrie | California State University, Fullerton | United States |
| Roman Pichler | Pichler Consulting Ltd | United Kingdom |
| Katherine Radeka | Whittier Consulting Group, Inc. | United States |
| Radhika Ramnath | Cadence Design Systems | India |
| John Shambroom | Shambroom Associates, LLC | United States |
| Kulbhushan Sharma | Quark Media House India Pvt. Ltd. | India |
| Thomas Sigemyr | IVF | Sweden |
| Reardon Smith NPDP, CMC | Business Vectors Inc. | United States |
| James Joseph Snyder | Medtronic, Inc. | United States |
| Gary J. Summers, Ph.D. | Skillful NPD | United States |
| John Tepper | Alpha Med-Surge, Inc. | United States |

Index

A

A3 reports, 113–114
adaptive action, 180
Agile Development Conference, 286
agile development style: adaptive action to correct plans in, 179–180; automating building and testing activities, 212; backlog vs. project completion in, 180; barely sufficient processes in, 209; coping with change in, 5; Extreme Programming, 14; feature replacement in, 33; loose-tight planning with iterations, 188–189; methodologies for, 14; pairing and, 144; process standardization with, 207; refactoring and technical debt, 226, 228; response to change vs. following plans, 179; robustness and agile concepts, 79; team meetings for, 135–136; using for other products, 25–26, 29; values of, 231–232; “you aren’t going to need it”, 228–230. *See also* Agile Manifesto; XP
Agile Manifesto: design simplicity and, 16, 22, 26; development processes in, 205; emphasis on individual in, 182; feedback in XP practices and, 23; individuals acknowledged in, 127, 152; methods outlined in, 13, 14; response to change vs. following plans, 179; text of, 12; value of working products, 211
Agile Project Leadership Network, 177, 286
Agile Project Management (Highsmith), 177, 190
Agile Retrospectives (Derby and Larsen), 201
analysis-think-change, 240
anticipating customer needs, 47–52;

anticipating change, 5–6; forecasting vs., 47; immersing developers in customer environment for, 48–51, 55; project planning vs., 185–186
Apple Computer, 50
assessing explorations, 91, 93–94
authority and teams, 139–141, 152
automobile design: body panel decisions in, 119; delaying decisions on, 118–120; using modular architecture, 65–66
Avery, Christopher, 138

B

backlog in agile development, 180, 188
Beck, Kent, 20, 23
Berra, Yogi, 107, 114, 177, 178
best practices: heavy process and, 208; management judgment and, 26–27; opposite of, xiii; product development and, 13; project difficulties following, 129–130
Black & Decker, 66–67, 73, 170–171
Black-Scholes formula, 174
blogs, 49–50
BMW, 41
bodies of knowledge approach, 129–130
Boehm, Barry, 8–10, 24, 35–36, 125–127, 131, 215–221
Boeing, 189
bottlenecks and queues, 221–226; identifying bottlenecks, 222–223; impeding flexibility with queues, 226, 230; theory of constraints methodology, 222; wait times and myth of capacity, 223–226
bottom-up change, 244–248; combining with top-down, 245; top-down vs., 233–234

- Box, George, 163
 brainstorming, 251–252
 Bridges, William, 237–238, 245
 Brooks, Frederick, 231
Building the Bridge As You Walk On It, 127
 building vs. scaling down processes, 207–209, 230
 burndown charts, 198–199
 burnout, 18
- C**
- calculus, 110
 call options, 173
 Callahan, Jim, 150
 Canadian pharmacy, 264
 cascading change, 117
 C-Cor, 150
 CD-ROM drive architecture, 67–72, 80–81
 change: accommodating predictable, 67–73, 84; agile skills coping with, 5; anticipating, 5–6, 249–264; cascading, 117; cost of, 8–10; dealing with product, 2–7; engineering, 107, 119; experience dealing with, 129; feeling at core of successful, 239–240; guiding via teams, 241; higher level requirements reducing, 40–47, 55; identifying bottlenecks with proposed, 223; individual's comfort with, 128; influences requiring project, 4–5; innovation and, 2, 29; institutionalizing, 244; intrinsic risk management with rapid, 192; isolating volatility, 74; maintaining continuous, 243; managing projects and, 177–178; providing for product growth, 73, 80; top-down, 239–244; transition vs., 237; urgency leading to, 240; user stories keeping cost low for, 46–47; using flexibility tools offensively, 264. *See also* cost of change; organizational change
- chicken test, 105
 Chrysler, 20, 43, 119
 chunk architectural elements, 57, 58–59, 60
 clustering product elements, 76
 Cockburn, Alistair: 144, 207, 212. *See also* mastery levels for team members
 COCOMO (Constructive Cost Model), 125
 Cohn, Mike, 156–157
 collective code ownership: coding standards and, 19–20; defined, 18; safety net for, 21; translating practice to other products, 26
 co-located teams: communication patterns for, 148–150; dispersed teams and, 141, 142; meeting together initially, 147; partially, 146–150; productivity and, 143–144; working with, 14, 141–146
 communications: analyzing co-located team, 148–150; cascading changes in corporate, 117; communicating organizational change, 241–242; cultural factors and, 22; managing team's electronic, 150–151, 152; as XP value, 20
 computers. *See* desktop computers
 conducting retrospectives, 202–203
 consensus in group decisions, 159–161
 Constantine, Larry, 57
 constraints: set-based design, 109–113; Toyota constraints checklists, 112
 constructing exploration models, 91, 92–93
 Constructive Cost Model (COCOMO), 125
 continuous integration, 18, 21, 26
 convergence, 10–11, 116
 cooked meat instrument, 218–221, 250–251
 cordless screwdrivers, 66–67, 170–171
 corporate culture. *See* cultural factors
 cost of change: factors influencing labor costs, 125–127; front-loaded prototyping and, 95; growing during project, 8–10; in hardware vs. software, 65, 83, 211; lowering with delayed decisions, 119; modular architecture and, 64, 67; psychological, 123; set-based design and, 114–115; simple design and, 16, 17, 22, 228–229
 costs of iterations: economics of prototyping, 101; lowering, 39–40; 3-D printer prototyping and, 97–100
 courage, 21, 23
 Credit Suisse, 49
 critical path, 156
 critical success factors, 2
 cross-functional teams, 138–139; product architecture and, 73; product requirements and, 34
 cultural factors, 232; communication and, 22; corporate cultures developing Level Xers, 131; corporate discouragement of failure, 90; impeding flexible development, 38; inertia and organizational change, 96; set-based design and Western mindset, 115
 currencies used, 275

customer iteration cycle. *See* iterations
 customers: anticipating needs of, 5–6,
 47–52, 55; changes in product require-
 ments by, 4; combining iterations
 with feedback from, 35–36, 37–38, 55;
 completion measured by feedback
 from, 180–181; developing product
 requirements from needs, 31; having
 on team, 26; IKIWISI, 4, 31; immers-
 ing developers in customers' environ-
 ment, 48–51, 55; incremental releases
 for, 209–210; internal, 54; lead user
 research, 51–52; misleading feedback
 from expert, 53; pitfalls of feedback
 from, 52–54, 55; users vs., 267; value of
 feedback in product development, 34–
 40; value of incremental development,
 210; vision statements developed from
 surveys of, 42; VOC research with,
 53–54

D

Darwin, Charles, 1
 DeCarlo, Doug, 177
 decision gradients, 159–160
 decision making, 153–176; choices for
 improving, 153–154; critical path
 in, 156; cross-functional, 138–139;
 decision gradients, 159–160; defer-
 ring, 108; finding linked decisions,
 171–173; imperfect nature of, 161; last
 responsible moment, 154–157, 175;
 mastery levels and quality of, 129–131;
 point-based vs. set-based, 109–111;
 preventing premature, 115, 118–122;
 procrastination vs. last responsible
 moment, 157–158; progressive, 120–122;
 pushing decisions down to improve
 response time, 194; real options
 thinking, 173–175; set-based design and
 better, 115; team structure and, 158–
 161; timeboxes and, 190; uncertainty
 and, 161–162, 165, 167, 192, 204. *See also*
 decision trees; delaying decisions
 decision tree software, 170–171
 decision trees, 162–173; about, 162–164,
 176; benefits of, 172–173, 176;
 employing, 164–169; finding linked
 decisions in, 171–173; flipping, 170–171;
 illustrated, 163, 166, 167, 168; value of
 perfect information in, 169–171

delaying decisions: accommodating change
 in cordless screwdriver, 67; better
 information in set-based design with,
 115; making last responsible moment
 decisions, 154–157, 175; photocopiers as
 example of, 120, 121; procrastination vs.,
 157–158; set-based design and deferring
 decisions, 108; Toyota's style of, 118–122

Denso, 66

Derby, Esther, 201, 203

Design of Experiments methodology, 101

design simplicity, 16, 22, 26

desktop computers: CD-ROM drive
 development for, 67–72, 80–81;
 designing interface of, 79; drawing
 architectural schematics of, 75–76;
 improvements in prototyping using
 technology of, 86; modular architecture
 in, 64–65

Developing Products in Half the Time (Smith
 and Reinertsen), xv, 121, 281, 285

development processes, 205–230;

balancing anticipation and adapta-
 tion, 212–213; bottlenecks and queues,
 221–226; build rather than scale
 down, 207–209; emergent, 205–209;
 flexible processes basics, 209–215; low-
 level standardization, 206–207, 230;
 managing change with YAGNI, 228–
 230; refactoring and technical debt, 226,
 228; risk balancing in, 215–221, 230;
 shifts from flexible to structured, 221,
 222, 230; software methodology research
 on, 208–209; structure balanced with
 flexibility, 215–221; tacit knowledge,
 213–215, 230; wait times and myth of
 capacity, 223–226

development teams. *See* teams

disclaimers, 26–27

dispersed teams: about, 141, 142; changes
 due to, 5; complications running,
 146–147

Disraeli, Benjamin, 249

disruption: flexibility and lack of, 2;
 systemwide impact of change and, 8, 9

Duarte, Deborah, 146, 150

E

Edison, Thomas, 85, 87
 e-mail standards, 150–151, 152
 emergence: process and, 127, 205–206;
 requirements and, 4, 38
 employee blogs, 49–50
 empowering teams, 234, 243
 Evangelists, 246
 examples, dated, xix. *See also* product scenarios
 exhaust systems, 118
 expectations management, 190–191
 experimentation, 85–106; corporate discouragement of failure, 90; defined, 85, 86–87; exploration as, 90–94; failure in, 87–90; flexibility and, 85–86; mistakes vs. failures, 88; observing experiments, 91, 93; prototyping and, 94–104; resolving uncertainty in decision making with, 162; technologies enabling prototyping, 97–98, 99; testing prototypes, 104–106
 expert customer feedback, 53
 explicit knowledge, 182, 213–214
 exploration: assessment step for, 91, 93–94; constructing models in, 91, 92–93; defined, 90–92; iterative process in, 91; managing risks with, 194; planning for, 91, 92; run step for, 91, 93
 Extreme Programming. *See* XP
eXtreme Project Management (DeCarlo), 177

F

failure: corporate discouragement of, 90; experimentation, 87–90; finding bigger problems ahead from, 193; learning from prototyping, 89; mistakes vs., 88; testing for product, 104–106
 Farnbach, John, xi–xii
 FDA (U.S. Food and Drug Administration), 32, 275
 feedback, 34–40; combining iterations with customers', 35–36, 37–38, 55; measuring product completion by, 180–181; pitfalls of expert customer, 52–54, 55; XP and, 21, 23
 feelings, 199; and change, 239–240
 Ferrari F430 sports car, 62, 63
 final testing, 106

flexibility: applying selectively, 9; benefits of, 7–8; calculus thinking limits, 110; convergence and, 10–11, 116; cultural factors impeding, 38; decision trees and, 162–163; definitions of, 1–2, 196–198; difficulties implementing set-based, 122–124; discretionary use of, 29; downside of, 11–12, 114–115; employed offensively, 264; enhancing with experimentation, 85–86; excessive, 10–11; importance of teams in, 127–128; lack of disruption and, 2; last responsible moment and, 154–157, 175; levels of project, 10–11; managing unknown risks and, 193–195; modular architecture and, 58–59, 63–64; organizational changes required for, 90, 96; paradoxes in implementing, 233–237, 248; postponement and, 59; project management as key in, 177–178; reducing costs of with set-based design, 114–115; strengthening tacit knowledge for, 182; XP practices and, 14. *See also* flexible processes; implementing flexibility; XP Flexibility Index, 196–198
 flexible processes, 205–215; about, 205–206; balancing risk in, 215–221, 230; emergence of, 205–209, 230; iterative and incremental innovation, 209–212, 230; low-level standardization in, 206–207, 230; queues as impediment to, 226; scaling down processes, 207–209, 230; shifting to structured processes, 221, 222, 230
 flexible project management: institutionalizing, 244; mainstream vs., 178, 203; manager's role in, 182–189; project plans in, 178–180; rolling-wave planning, 186–188
 flipping decision trees, 170–171
 food processor, 259–263
 footstool, 42, 241, 242
 following plans: allegiance to, 2, 178–180; response to change vs., 179; rewards for, 181
 forecasting, 6, 47
 front-loaded prototyping: guidelines for, 100–104; traditional vs., 95–96; using for surgical laser, 98–100
 functional architectural elements, 57, 58–59, 60

G

Gates, Bill, 31
 generalists, 136–137
 Generation X lifestyle, 50
 Google, 50
 Griffiths, Mike, 147
 Grove, Andrew, 194–195

H

heating pad decisions, 167–169
 Hewlett, Bill, 183, 240
 Hewlett-Packard, 41–42, 195–196, 240,
 241, 242
 Highsmith, Jim, 54, 177, 190
 Hohmann, Luke, 48

I

Iansiti, Marco, 91
 IBM, 88
 IKIWISI (“I’ll know it when I see it”), 4, 31
 IMNIL (“I might need it later”), 229
 imperfect decision making, 161
 implementing flexibility, 231–248; bottom-
 up change, 233–234, 244–248; building
 vision of change, 241; celebrating
 successes in, 243; empowerment
 and, 234, 243; exposing or sheltering
 projects, 237; gradual or ambitious
 goals for, 236–237; guiding change via
 teams, 241; paradoxes in, 233–237, 248;
 starting small or big, 234–235; starting
 with piece or whole package, 235–236;
 summary, 248; top-down change, 233–
 234, 239–244; transitions in, 237–239.
See also flexible project management
 improving decision making, 153–154
 incremental innovation, 209–212, 230
 indecision and flexibility, 11–12
 inefficiencies in over-dedication, 133–135
 information technology (IT): 213–214, 215;
 customers and, 19, 180
 innovation: areas where flexibility can
 aid, 7–8; change required for, 2, 29;
 decline in, 2–3; iterative and incre-
 mental, 209–212, 230. *See also* product
 development
 integral product architecture: defined, 59;
 modular vs., 58–59, 63; performance
 advantages, 60–61, 63; shifting design
 from modular to, 61. *See also* modular
 product architecture

integrated risk management, 191–192
 Intel, 131, 194
 interaction of product modules, 80
 internal customers, 54
 intrinsic risk management, 191–192
 ISO 9000, 3
 iterations: centrality to flexibility, xix,
 combining with customer feedback, 35–
 36, 37–38, 55; exploration and iterative
 process, 91; feature replacements
 during, 33–34; having working product
 at end of, 14; incremental innovation
 processes and, 209–212, 230; iterative
 organizational change, 234, 235; loose-
 tight planning and, 188–189; lowering
 costs of, 39–40; managing risks with,
 194; measuring completion in, 180–181;
 retrospectives for, 202; risk manage-
 ment and, 192–193; story cards for,
 46; velocity of, 189. *See also* costs of
 iterations

J

Jaguar, 41
 jeepney, 61–63
 Johnson & Johnson Worldwide Emerging
 Markets Innovation Center, 26

K

Kerth, Norman, 201
 Kessler, Robert, 144–145, 146
 keypad, 81–82
 knowledge management, 213
 Kotter, John, 234, 239–243, 245, 248

L

language layers, 206–207
 Larman, Craig, 12
 Larsen, Diana, 201, 203
 last responsible moment: defined, 155;
 procrastination vs., 157–158; using,
 154–158, 175
 lead users, 51–52, 53
 leadership for implementing flexibility,
 233–234
 leaf nodes in decision trees, 164
 Level X developers, 130, 131, 145, 159

levels: project flexibility, 10–11. *See also*
 mastery levels
 Lexus, 41, 42–43
 Liker, Jeffrey, 200
 linked decisions, 171–173
 lock analogy, 102–104
 loose-tight planning, 188–189
 low-level standardization, 206–207

M

M/M/1/∞ queue, 224, 225, 226
 MacCormack, Alan, 34–36, 91, 128–129, 181,
 215, 220
 mainstream project management: about,
 177; allegiance to project plans, 178–180;
 emphasis on processes in, 182; flexible
 vs., 178, 203; views of project completion,
 180–181
 management by walking around (MBWA),
 183–184
Managing the Design Factory (Donald
 Reinertsen), 140
 Manns, Mary Lynn, 245, 247–248
 manufacturing, 1
 market changes, 4
 Marshall, Jon, 155
 mass customization, 1, 59
 mastery levels for team members: Cockburn,
 129–130, 131–132, 206, 208, 209, 214. *See*
also Level X developers
 McQuillen, David, 49
 Meat temperature sensor, 218, 250–251
 Mercedes Benz, 41, 43
 methodologies: agile software development,
 14; Design of Experiments, 101;
 development processes in software,
 208–209; Pugh concept selection, 112;
 theory of constraints, 222. *See also* agile
 development style; Agile Manifesto
 metrics: burndown charts, 198–199;
 Flexibility Index, 196–198; sharing and
 acting on, 200; strategic vs. tactical,
 195–196; team mood as, 199–200
 Microsoft, 49, 73
 mindfulness, 194
 mistakes: exploration vs., 90; failures vs., 88
 modular product architecture, 57–84;
 aligning organization boundaries with,
 74–75; automobile design using, 62,
 65–66; CD-ROM drive, 67–72, 80–81;
 cordless screwdrivers, 66–67;

decisions in approaches to, 72–74;
 defined, 58–59; designing interface,
 79; desktop computers' use of, 64–65;
 isolating volatility, 73; objectives for,
 63–64; placing functionality in modules,
 78; providing for product growth, 73,
 81; reducing coupling, 73–74; shifting
 boundaries in architectural domains,
 83; steps in designing, 74–78. *See also*
 integral product architecture
 money-for-adaptability (MFA), 212, 213
 money-for-information (MFI), 212, 213
 Morgan, James, 200
 Mountain Goat Software, 156–157
 mules, 40, 93

N

NASA (National Aeronautics and Space
 Administration), 218
 needle-in-a-haystack metaphor, 6

O

OEM (original equipment manufacturer),
 169, 219
 offensive flexibility, 264
 Olson, Gary, 143, 146
 Olson, Judith, 143, 146
Only the Paranoid Survive (Grove), 194–195
 organizational change: bottom-up,
 233–234, 244–248; building vision of,
 241; celebrating successes, 243; com-
 bining bottom-up and top-down, 245;
 communicating and building owner-
 ship in, 241–242; continuing, 243;
 empowerment and, 234, 243; exposing
 or sheltering projects, 237; gradual or
 ambitious goals for, 236–237; guiding
 via a team, 241; leadership required for,
 233–234; overview, 232–233; patterns for,
 246–248; risks in iterative, 235; starting
 small or big, 234–235; starting with piece
 or whole package, 235–236; top-down,
 233–234, 239–244, 245. *See also* analysis-
 think-change, see-feel-change
 Orion, 98–100

P

Packard, Dave, 183
 pair programming. *See* pairing

- pairing, 17–18; agile software development and, 144; coding standards, 19–20; continuous integration with, 18, 21, 26; non-software example, 145–146; origins of, 20; safety net for collective code ownership, 21; translating practice to other products, 26; types of, 145
- parallel development: parallel prototyping, 102–104; set-based design vs., 112–113
- Pareto principle, 8
- patterns: defined, 245; organizational change using, 246–248
- people factors: commitment to team, 135–136; cultivating specific areas of depth, 137; dedication of time, 133–135; desirable qualities for teams, 132–137; establishing paths for mastery, 132; evaluating mastery levels, 129–131; importance of feelings in change, 239–240; individual's comfort with change, 128; influence on development costs, 125–126, 132; organizational change and, 237–238; phases in transition process, 235; technical and social skills, 132–133; types of pairing by, 145; useful experiences, 128–129. *See also* cultural factors
- personal computers. *See* desktop computers
- personas, 44
- Peters, Tom, 105
- phased development: xvi, 3–4, 5–6, 10; decisions and, 153, 154–155; documentation heavy, 32, 211; iteration and, 193, 209; layers of development process and, 207
- photocopiers, 120, 121
- pig, chicken, and cow analogy, 135–136
- planning: anticipation vs., 185–186; exploration process, 91, 92; loose-tight, 188–189; need for 38; rolling-wave, 186–188; XP project, 15. *See also* project plans
- platform architecture, 59
- platforms, 1
- point-based design, 108–110, 111, 113
- postpartums/postmortems, *See* retrospectives
- postponement, 59
- printers; 48; 3-D, 97–98, 99; Hewlett-Packard DeskJet, 41–42, 241, 242
- probabilities, estimating, 254
- problem-solving, 129, 244
- processes. *See* development processes
- procrastination: anticipation vs., 186; last responsible moment vs., 157–158
- product architecture: attention to, 34; defining, 57–58; design-level changes in, 81–83; modular vs. integral, 58–63. *See also* integral product architecture; modular product architecture
- product audit, 201–202
- product development: creating working products early in cycle, 34; customer feedback in, 34–40; decision making objectives and methods for, 153–154; designing to specification, 31–32; factors influencing costs of, 125–127; implementing flexibility in small or big projects, 234–235; loose-tight planning with iterations, 188–189; overspecification trap in, 36–37; prototyping and successful, 35–36; requirements creep, 32–34; specifying at higher level, 40–47, 55. *See also* innovation; project management; specifications
- product elements: chunks and functional, 57, 58–59, 60; clustering, 76
- product vision: project manager's preservation of, 184; statements of, 41–44
- productivity and co-location, 17–18; 143–144
- products: accommodating predictable change for, 67–73, 84; changes to improve reliability, 240–241; designing interface for, 79; drawing architectural schematics of, 74–76; emergence of requirements, 4, 38, 127; isolating volatility in, 74; metaphor used in XP, 16; objectives for modular, 63–64; pitfalls of customer feedback on, 52–54, 55; placing functionality in modules, 78; project personas for, 44–45; providing for growth of, 73–74, 81; reorienting quality of, 181–182; unused features in, 37; vision statements for, 41–44; working, 13, 30, 34–35, 188, 209, 211. *See also* modular product architecture
- progressive decision making, 120–122
- Project Analyzer, 27–29
- project management, 177–204; aligning organization with product architecture, 75; allegiance to project plans,

- 178–180; applying to innovative projects, 4; authority of teams, 139–141; burndown charts, 198–199; Flexibility Index, 196–198; flexible vs. mainstream, 178, 203; manager's role in flexible, 182–184; managing set-based design, 115–117; measuring team mood, 199–200; planning vs. anticipation, 185–186; project completion, 38; reorienting product quality, 181–182; risk management, 191–195; sharing and acting on metrics, 200; shifting emphasis from processes to individuals, 182; strategic vs. tactical metrics for, 195–196; tacit vs. explicit knowledge, 182, 213–214; timeboxing, 190–191; using retrospectives in, 200–203; working with Level Xers, 131. *See also* planning; project managers; project plans
- Project Management Institute, 177, 191, 251
- project management software, 179
- project managers: managing by walking around, 183–184; preserving product vision, 184; role in flexible project management, 182–184; supporting and shielding team members, 184
- project plans: allegiance to, 2, 178–180; measuring success and rewards by, 181; response to change vs., 179
- Project Retrospectives* (Kerth), 201
- project risk management, 191–195; about, 191; integrated vs. intrinsic, 191–192; iterative development and, 192–193; managing unknown risks, 193–195
- projects: establishing thumbprints for, 27–29; evaluating value of undertaking, 174; exposing or sheltering transitional, 237; Flexibility Index for, 196–198; maintaining sustainable pace, 18; process shift over life of, 221, 222, 230; tracking completion with burndown charts, 199; views of project completion, 180–181
- prototyping: defined, 94; delaying decisions during automotive, 118–120; economics of, 101; experimentation and, 94–104; learning from failures in, 89; lowering costs of, 39–40; MacCormack's findings on, 34–35; parallel vs. sequential, 102–104; refinement of, 100–101; technologies enabling, 97–98, 99; traditional vs. front-loaded, 95–96; user feedback and, 35–36; pruning, 115–118
- Pugh concept selection method, 112
- put options, 173
- ## Q
- Quadrus Development, 147, 264
- queues: impeding flexibility with, 226, 230; M/M/1/∞ model for, 224, 225, 226; wait times in, 224, 225, 226, 227
- queuing theory, 224
- ## R
- radical collocation, 143
- rapid prototyping, 96–100, 275
- reading decision trees, 164
- real options thinking, 173–175
- reducing coupling, 73–74
- refactoring: about, 17, 226, 228; arresting architectural decay and, 80; safety net for collective code ownership, 21
- refinement of prototypes, 100–101
- Reinertsen, Donald, 32, 34, 121, 140, 142
- reliability of products, 240–241
- requirements: emergent, 4, 38, 127; requirements creep, 32–34. *See also* specifications
- retrospectives, 200–203; benefits of, 200–201; conducting, 202–203; iteration, 202; postmortems and, 201; project, 202
- rice cooker use case, 45–47
- Rising, Linda, 245, 247–248
- risks: balancing opposing, 215–221, 230; decision making triggered by, 156; found in iterative organizational change, 235; loose-tight planning and managing, 189; organizational change and, 234. *See also* project risk management
- robustness, 79
- rolling-wave planning, 186–188
- Rolls-Royce, 105
- ## S
- scaffolding, 40, 93, 105
- scaling down processes, 207–209, 230
- schematics, 57–58, 70–71, 75–76
- Schrader tire valve, 80

- scope creep, 32–34
 see-feel-change, 240, 241, 248
 selective flexibility, 9
 self-organizing teams, 25, 138
 sequential prototyping, 102–104
 set-based design, 107–124; benefits of, 114–115; constraints basis, 109–113; defined, 108; delaying decisions for better information, 115, 118–122; difficulties implementing, 122–124; point-based vs., 108–110; problem solving with, 111–113; pruning, 115–118; technical reports supporting, 113–114
 single-minute exchange of die (SMED), 1
 Six Sigma, 3, 5
 sketching design layout, 76–78
 skills: cultivating specific areas of depth, 137; establishing paths for gaining mastery and, 132; evaluating mastery levels, 129–131; specialists vs. generalists, 136–137; technical and social, 132–133; types of pairing by, 145
 small releases, 15–16, 26
 Smith, Preston G., xvi, 121, 285–286
 Snyder, Nancy, 146, 150
 social skills, 132–133
 software, special characteristics of, 25
 Sony Walkman, 60, 61
 specifications: avoiding change with higher level, 40–47, 55; designing product to, 31–32; development using prototyping vs., 34–36; excessive, 36–37; frozen, 31–32, 54–55; vs. requirements, 253
 Spinrad, Robert, 153, 154
 Standish Group, 37
 stand-up meetings: agilists use of, 135–136; formats for, 148–150
 stereolithography (SL), 97
 story cards for iterations, 46
 Stage-Gate. *See* phased development
 strategic project metrics, 195–196
 structured development processes: risks in, 216, 217; shifting from flexible to, 221, 222, 230
 stubs, 93, 105
 success: celebrating implementation, 243; commercial success vs. manager's view, 34–35; feelings at core of successful change, 239–240; improving chance of in implementation, 235; innovation and critical success factors, 2; measuring by project plan, 181; prototyping products, 35–36
 surgical drapes, 52
 surgical laser, 98–100
 Suzuki, Ichiro, 42
 synergy in whole-package transitions, 235–236
- ## T
- tacit knowledge, 182, 213–215, 230
 tactical project metrics, 195–196
 TEAC, 67–71
 team leaders. *See* project managers
 teams, 125–152; authority of, 139–141, 152; changes facing dispersed, 5; co-located, 14, 141–146; commitment and pig, chicken, and cow analogy, 135–136; creating great, 131–132; cross-functional, 138–139; decision making within, 158–161; dedicating people to projects, 133–135; desirable human qualities on, 132–137; difficulties with Level X members, 130, 131, 145, 152; electronic communications among, 150–151, 152; empowering to lead change, 234, 243; experience in product delivery, 34–35; factors influencing development costs, 125–127; guiding change via, 241; having customer on, 18–19; importance in flexibility, 127–128; managing by walking around, 183–184; measuring mood of, 199–200; meetings for agilists, 135–136, 148–150; partially co-located, 146–150; roles and responsibilities of members, 138; selecting people on, 128–132; self-organizing, 138; sharing metrics with, 200; specialists vs. generalists on, 136–137; subteams for set-based design, 123; supporting and shielding members of, 184; sustaining project pace, 18; training in flexible development techniques, 242. *See also* co-located teams; dispersed teams
Teamwork Is an Individual Skill (Avery), 138
 technical debt, 228
 technology: change stemming from, 4; changes in, 5; choosing with decision trees, 163–164; enabling shifting boundaries in architectural designs, 83; prototyping enabled by, 97–98, 99
 template for a change, 250–251

test-driven design, 16–17, 21, 26
 testing: deferring when expensive, 211–212;
 front-loaded vs. final product, 106; as
 experimentation, 104–106; value of
 failure in, 87–90
 theory of constraints, 222
 Thomke, Stefan, 88, 100
 3-D printers, 97–98, 99
 3M surgical drapes, 52
 thumbprints for projects, 27–29
 timeboxing, 190–191
 tolerances, 119–120
 tool kit metaphor, xvii, 232
 tool-safe dies, 122
 top-down change: bottom-up vs., 233–234;
 combining with bottom-up, 245;
 Kotter's steps for, 239–244
 tortillas, 48
 Toyota: constraint checklists, 112; design
 of interfaces at, 79; immersing devel-
 opers in U.S. environment, 50; isolating
 non-changing areas, 65–66; lack of
 employee complacency at, 243; man-
 aging rate of convergence, 116; pre-
 venting premature decisions, 122, 123;
 product vision statement at, 43; retro-
 spectives at, 200; SMED technique at,
 1; standardizing low-level activities at,
 207; tradeoffs described in $\Delta 3$ reports,
 113–114; use of set-based design, 107;
 use of tacit knowledge, 214, 215
 trade-off curves, 112
 training, 242
 transitions, 237–239
 T-shaped individuals, 137
 Turner, Richard, 24, 125, 131, 215

U

uncertainty: determining sensitivity of
 decisions to, 165, 167; intrinsic risk
 management and, 192, 204; reducing
 decision making, 161–162
 underlying XP values, 20–23
 U.S. Army jeep, 61, 62, 63
 U.S. Food and Drug Administration. *See*
 FDA
 U.S. National Aeronautics and Space
 Administration (NASA), 218

unk unk (unknown unknown) risks,
 193–195
 urgency for change, 240
 use case technique, 45–46
 user stories, 46–47

V

values in agile development style, 20–23,
 231–232
 velocity, 189
 Venn diagrams, 112, 113
 virtual teams. *See* dispersed teams
 vision: building for organizational change,
 241, 242; product, 41–44, 184
 voice of the customer (VOC), 53–54

W

waterfall processes, 181, 193, 201
 Watson, Sr., Thomas, 88
 Williams, Laurie, 144, 146
 Windows XP, 73
 Wohlers Associates, Inc., 97

X

XP (Extreme Programming): about, 14–15;
 applied to personal care products, 26;
 assessment of, 23–25; coding standards,
 19–20; collective code ownership, 18,
 19–20, 21; continuous integration with,
 18, 21, 26; design simplicity in, 16, 22,
 26; feedback valued as practice, 21,
 23; having customer on team, 18–19;
 lowering costs of iterations, 39–40;
 maintaining sustainable pace in, 18;
 origins of, 20; pairing, 17–18; product
 metaphor in, 16; project planning in,
 15; refactoring, 17; small releases, 15–16;
 test-driven design, 16–17, 21; underlying
 values of, 20–23. *See also* iterations;
 pairing; refactoring

Y

YAGNI (“you aren’t going to need it”),
 228–230
 Yahoo!, 50
 Young, John, 240, 241, 243