Praise for *Agile IT Organization Design*

“Continuous delivery is often described from the perspective of the technicians. This is understandable because that is where it started, but it does the process a disservice. Continuous delivery is a holistic approach. It requires change across the organization and it encourages such change, to the betterment of the groups that practice it. This book addresses that problem and looks at CD from an organizational perspective. It starts from Dan Pink’s ideas of intrinsic and extrinsic motivators and describes how to structure an organization for success—how to encourage a focus on autonomy, mastery, and purpose that will motivate your teams and produce high-quality results. This book takes a look at all aspects of organizational design that impact the ability to deliver regular, small, high quality changes. If you follow the advice in this book, your organization will be the better for it.”

—*Dave Farley, author of Continuous Delivery*

“A number of years ago, Silicon Valley marketing guru Geoffrey Moore quipped, ‘A bank is just a computer with a marketing department.’ Today, technologies—cloud, social, big data, the Internet of Things, and mobile—continue to drive this unprecedented digital transformation in organizations. As such, the need for agility has moved from software development to corporate boardrooms. Sriram’s book makes the case that to thrive in these fast and uncertain times, enterprise leaders need to rethink how IT, not just software development, is organized, structured, and measured. His book provides guidelines, not prescriptions, which enable innovation, adaptability, and responsiveness at scale.”

—*Jim Highsmith, Executive Consultant, ThoughtWorks, Author of Adaptive Leadership*

“Very hands-on and operational book for management of Agile-based development. Provides valuable insight for IT practitioners. A must read for IT professionals.”

—*A.V. Sridhar, Founder, President & CEO Digite, Inc.*

“Agile IT Organization Design is an engaging, enlightening, and immensely practical book. While many authors have addressed Agile software development, very few have tackled the wider topic of the more systemic changes necessary to move from Agile software to an agile organization, and onwards to ‘digital transformation.’ Even fewer have done so at more than a very theoretical level. Drawing heavily upon his substantial practical experience, Sriram Narayan’s
book explores the pitfalls of many of our current ‘organizational wisdoms’ and gently, but convincingly, suggests appropriate and relevant alternatives to try in their place—all the time backed up by real-world examples. I highly recommend the book to anyone interested in, or struggling with, the challenges and opportunities of achieving organizational agility.”

—Chris Murphy, President and Chief Strategy Officer, ThoughtWorks

“Agile and continuous delivery transformations require changes in technology, process, and people. This book is the first to tackle the people aspect in depth, and it does this very well. A must read for those taking the journey!”

—Anders Wallgren, CTO, Electric Cloud

“Agile IT Organization Design tackles all the problems that we just want to ignore. Relying heavily on hands-on experience rather than theoretical exercises, Sriram provides concrete actions to address the issues with Agile software development and continuous delivery at a structural and organizational level. He clearly addresses issues of finance, accountability, and metrics, not just team structure and team processes, and gives many examples and scenarios to help understand how these issues manifest and how the proposed steps work to resolve the issues. Organizational transformations to Agile often fail, not because the individual processes and practices break down, but because the organization itself—its power structure, its organizational norms, and its culture—fight against the gains that Agile has the potential to bring. Sriram focuses our attention on the systemic problems, but then provides action steps to allow us to address these problems in our context. This book presents no silver bullet, as those don’t exist. However, Sriram provides for organizations a way to start facing reality and moving towards an organization that supports not only Agile software development but organizational and business agility.”

—Rebecca Parsons, Director at Agile Alliance & CTO at ThoughtWorks

“Sriram’s book addresses the rarely-approached topic of Agile organization design in a very pragmatic and thorough manner. It does a great job of explaining the value brought by Agile and DevOps approaches in enterprise-scale organizations, and gives strong details on the ‘how’ to get there. It also paints a very practical picture of how the different processes of the company (budgeting, staffing, metrics, etc.) will be affected by the Agile organizational choices. I see it as the perfect companion book for a large-scale Agile transformation effort.”

—Regis Allegre, VP Software Engineering, Cloudwatt
“Businesses today are discovering that if they are to build ‘digital first’ experiences for their customers, they need to rethink how their product, marketing, and technology teams work together. Sriram’s book pulls aside the curtain to reveal that the best-kept secrets of the world’s top performing digital organizations are actually very accessible to all. It serves as a pattern language for management of the modern digital enterprise.”

—Adam Monago, VP Digital Strategy, ThoughtWorks, @adammonago

“Agility is so much more than stand-ups and test driven development. Even the best practices won’t yield results unless backed by the right leadership. Sriram’s book is an important contribution to the all-too-bare bookshelf on leadership of IT organizations. He mixes theory and practical insights in the right measures and the result is as readable as it is full of usable insights.”

—Nagarjun Kandukuru, VP Global South Strategy, ThoughtWorks

“Sriram covers everything the Scrum coach didn’t tell you. Most books on Agile stop at a team and project level, and that’s exactly where the organizations tend to get lost in the real world of pre-existing organization structures and procedures—which in turn become blockers to achieving ultimate business agility. If you ever wonder why your attempt at Agile is floundering, this is one book where you’ll find some answers for sure.”

—Puneet Kataria, Vice President Global Sales, Kayako

“The field of Agile is an evolving, moving target and there is little in terms of guidance for managers and staff that are trying to implement it within an enterprise context. This book provides a complete guide to all of the organizational aspects of implementing Agile within the enterprise context, as well as providing extremely useful examples and cogent advice. I would recommend this book to anyone with a general interest in Agile through to senior managers looking to reenergize their enterprise organizations using the principles and practices of Agile.”

—Ken Robson, Global Head of Trading Technology, Danske Bank

“Sriram has pulled off an audacious attempt at a unified theory of IT. This work led me through the incredible range of issues that I recognize, slotting each one into context and building a vision of how things can and should be. If you want to be elevated above the trenches of Agile and DevOps—to get a better view of where they fit in the digital world that includes sales, finance, governance, resourcing, delivery, and most importantly, people—then read this book. A compelling read that I’m already referring back to.”

—Duncan Freke, Development Director, thetrainline.com
“Sriram makes a convincing case that digital transformation efforts need IT agility. He also does a great job of explaining how IT agility is more than just engineering and process. This book is a valuable read for those on the digital transformation journey.”

—Shashank Saxena, Director, Digital and eCommerce Technology, The Kroger Co.

“Adopting Agile software development practices is not just an IT change, it is an organization-wide change. Sriram goes through every aspect of what this means to an organization and gives options for how to bring changes in, including hard-to-change areas like project funding. This book is thought provoking, an easy read, and includes great examples.”

—Jeff Nicholas, Director, PB & WM IT Digital Banking APAC, Credit Suisse

“This book is for anyone who is looking for clear and focused guidance in the pursuit of modern product delivery. Any transformational leader will find this book a great tool that provides answers to many of the problems of Agile transformation at scale. A great jump start for those looking to improve their effectiveness and responsiveness to business, Sriram’s book recognises that people leadership is the DNA of any Agile transformation.”

—Marcus Campbell, Delivery Director, Semantico

“Entrepreneurial organizations thrive on continuously adding value, rapidly innovating, and staying close to their customers. Similarly, Agile software development emphasizes continuous, incremental improvements, quick response to change, and close collaboration. Sriram makes a compelling case for Agile design of IT organizations in large enterprises. He goes well beyond describing how an IT organization can adopt Agile development methodologies to explain how any successful digital transformation within a large enterprise must encompass strategy alignment, project portfolios, IT staffing, budgeting, and more. This book is a great read for those who want a digital transformation to have impact both within and beyond their enterprise IT organization.”

—Ron Pankiewicz, Technology Director, VillageReach

“Organizational structure is a key enabler for a company to achieve its raison d’être. This book lays out the rationale for organizing IT organizations around Agile software development concepts. It provides practical guidance on wide-ranging success factors including tangible org elements such as structure, team design, and accountability, and intangible cultural elements such as alignments and norms. These concepts will certainly help IT companies turn the tide on huge cost and time overruns that are typical on large IT projects.”

—Paul Kagoo, Engagement Manager at McKinsey & Co.
“Outcomes matter in an increasingly ‘winner takes all’ digital arena. A true
digital transformation undertaking, driven by the need to build competitive
advantage, is marked by an increase in responsiveness, insights, and engagement,
not just cost effectiveness. IT organization is a key partner in this transformation
but is seldom structured to succeed in most enterprises. This book makes a case
for how IT organization needs to be weaved within outcome-based teams, not
activity-based teams, to drive agility and competitive advantage. In general,
organizational design is very expensive to engineer in real world situations but
this book takes on this tough problem by providing some frameworks and
considerations for the reader to evaluate the validity of outcome-based structure
in their organization.”

—Vijay Iyer, Sr. Product Manager, NetApp

“I found Agile IT Organization Design to be well organized with an in-depth
knowledge of challenges that IT organizations face, while providing possible
ways to address those challenges. Moreover, it was eminently readable and I
found myself readily recognizing the problems described within. It may seem
odd to describe a business-oriented book as such, but I found this to be an
enjoyable read!”

—Randy R. Gore, Program Manager, IBM

“As enterprises try to ramp up their digital transformation initiatives, there will
be an ever-increasing need for better collaboration between IT and business.
New org structures will fuel this collaboration. Sriram’s book is a timely
elaboration of the importance of org structures for the success of digital
initiatives large and small.”

—Dinesh Tantri, Digital Strategist, @dineshtantri
This page intentionally left blank
Agile IT Organization
Design
This page intentionally left blank
For dear departed Amma and Appa
&
For Swati—my wonderful wife and elixir of my life
Contents

Preface ......................................................... xix
Acknowledgments ........................................... xxiii
About the Author ........................................... xxv
Glossary ....................................................... xxvii

Chapter 1: Context .......................................... 1
  1.1 Focus ................................................ 2
  1.2 Business, IT, and Shadow IT ...................... 3
  1.3 Business-IT Effectiveness ......................... 5
  1.4 Digital Transformation ............................ 7
  1.5 Bimodal IT and Dual Operating Systems ........ 10
  1.6 Angles of Coverage ................................ 10
  1.7 Summary ........................................... 11

Chapter 2: The Agile Credo ................................. 13
  2.1 Understanding the Agile Manifesto .............. 14
  2.2 Continuous Delivery and DevOps ............... 15
  2.3 Agile Culture ..................................... 16
  2.4 Common Themes ................................... 18
  2.5 Isn’t Agile Dead? .................................. 21
  2.6 Summary .......................................... 22

Chapter 3: Key Themes ..................................... 25
  3.1 Software Development Reconsidered ............ 26
  3.2 Govern for Value over Predictability .......... 28
  3.3 Organize for Responsiveness over Cost-efficiency .... 30
  3.4 Design for Intrinsic Motivation and Unscripted
      Collaboration ................................... 33
  3.5 Summary .......................................... 35
<table>
<thead>
<tr>
<th>Chapter 8: Projects</th>
<th>109</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 What Is Wrong with Plan-driven Software Projects?</td>
<td>109</td>
</tr>
<tr>
<td>8.2 Budget for Capacity, Not for Projects</td>
<td>110</td>
</tr>
<tr>
<td>8.3 Business-capability-centric IT</td>
<td>112</td>
</tr>
<tr>
<td>8.4 Project Business Cases</td>
<td>115</td>
</tr>
<tr>
<td>8.5 Value-driven Projects</td>
<td>117</td>
</tr>
<tr>
<td>8.6 Project Managers</td>
<td>119</td>
</tr>
<tr>
<td>8.7 Governance</td>
<td>120</td>
</tr>
<tr>
<td>8.8 Change Programs and Initiatives</td>
<td>121</td>
</tr>
<tr>
<td>8.9 Summary of Insights</td>
<td>123</td>
</tr>
<tr>
<td>8.10 Summary of Actions</td>
<td>123</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 9: Finance</th>
<th>125</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Relevance</td>
<td>125</td>
</tr>
<tr>
<td>9.2 Cost Center or Profit Center</td>
<td>126</td>
</tr>
<tr>
<td>9.3 Chargebacks</td>
<td>126</td>
</tr>
<tr>
<td>9.4 CapEx and OpEx</td>
<td>127</td>
</tr>
<tr>
<td>9.5 Conventional Budgeting</td>
<td>130</td>
</tr>
<tr>
<td>9.6 Agile Budgeting</td>
<td>132</td>
</tr>
<tr>
<td>9.7 Summary of Insights</td>
<td>134</td>
</tr>
<tr>
<td>9.8 Summary of Actions</td>
<td>135</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 10: Staffing</th>
<th>137</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Dealing with the Talent Crunch</td>
<td>137</td>
</tr>
<tr>
<td>10.2 Go Beyond Project Teams</td>
<td>139</td>
</tr>
<tr>
<td>10.3 Better Staffing</td>
<td>141</td>
</tr>
<tr>
<td>10.4 Summary of Insights</td>
<td>146</td>
</tr>
<tr>
<td>10.5 Summary of Actions</td>
<td>147</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11: Tooling</th>
<th>149</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1 Access Control for Unscripted Collaboration</td>
<td>149</td>
</tr>
<tr>
<td>11.2 Subtle Effects of the Toolchain</td>
<td>151</td>
</tr>
<tr>
<td>11.3 Technology Isn’t Value Neutral</td>
<td>154</td>
</tr>
<tr>
<td>11.4 Tool Evaluation</td>
<td>157</td>
</tr>
<tr>
<td>11.5 Summary of Insights</td>
<td>158</td>
</tr>
<tr>
<td>11.6 Summary of Actions</td>
<td>158</td>
</tr>
</tbody>
</table>
## Contents

Chapter 12: Metrics ......................................................... 159  
  12.1 Metrics Don’t Tell the Whole Story .......................... 159  
  12.2 Dashboards Promote Ignorance ............................... 162  
  12.3 The Problem with Targets and Incentives .................. 163  
  12.4 Reforming the Metrics Regime ............................... 171  
  12.5 Designing Better Metrics ..................................... 175  
  12.6 Objections to Metrics Reform ................................. 178  
  12.7 Migration ......................................................... 179  
  12.8 Summary of Insights ........................................... 180  
  12.9 Summary of Actions ............................................ 181  

Chapter 13: Norms ............................................................ 183  
  13.1 What Are Norms? ................................................ 183  
  13.2 Reinforcing Norms ............................................... 184  
  13.3 Cooperation over Competition .................................. 186  
  13.4 Living Policies .................................................. 187  
  13.5 Consistency over Uniformity .................................... 189  
  13.6 Ask for Forgiveness, Not for Permission .................... 192  
  13.7 Confidential Surveys ............................................ 193  
  13.8 Balance Theory and Practice ................................... 193  
  13.9 Summary of Insights ............................................ 195  
  13.10 Summary of Actions ............................................ 195  

Chapter 14: Communications ............................................... 197  
  14.1 Intrinsic Motivation ............................................. 197  
  14.2 Interpersonal Communications: Problems .................... 198  
  14.3 Interpersonal Communications: Mitigation ................... 203  
  14.4 Scaling Employee Engagement through Internal  
       Communications ................................................... 204  
  14.5 Deliberating in Writing ........................................ 208  
  14.6 The Use and Misuse of Visual Aids ............................ 211  
  14.7 Documents, Reports, and Templates .......................... 216  
  14.8 Summary of Insights ............................................ 217  
  14.9 Summary of Actions ............................................ 217  

Chapter 15: The Office ...................................................... 219  
  15.1 Open-plan Layouts .............................................. 219  
  15.2 Ergonomics ....................................................... 222  
  15.3 Remote Working ................................................ 224
Contents

15.4 Summary of Insights ........................................... 225
15.5 Summary of Actions ........................................... 225

Chapter 16: Wrap-up .............................................. 227
16.1 Summary of Effects ........................................... 227
16.2 Order of Adoption ........................................... 233
16.3 Information Radiators ......................................... 234
16.4 Sample Exercise ............................................. 235
16.5 IT Services ................................................... 236
16.6 GICs ........................................................... 240
16.7 Beyond IT ...................................................... 243

Bibliography ......................................................... 245
Index ............................................................. 247
This page intentionally left blank
Enterprise IT has mostly underperformed. It’s been a struggle to deliver IT-as-enabler, to say nothing of IT-as-differentiator. Partly as a result, it is common to hear of strained relationships between business and IT. This doesn’t augur well for digital transformation initiatives. I submit that a prime reason for the sorry state of affairs is poor organization design. A June 2014 McKinsey Global Survey also bears this out. It found at larger companies that structural issues are viewed as the top hurdle to meeting digital goals.

Organization design is traditionally considered from an industry-agnostic point of view. Although perceived to be the domain of HR, this is rarely the case when planning re-orgs. In contrast, this book explores organization design by IT leadership for IT organizations. It aims to provide a sound basis for IT organization design. This is essential because, in practice, IT organization design is rarely thought out from a baseline of principles. The prevailing design is mostly a product of happenstance, mergers or acquisitions, people-retention compulsions, and the ideas of whoever is in charge at various levels in the organization. It resembles how software accumulates technical debt over time unless we periodically step back and reassess the design.

Organization design in the 21st century is not merely structural but also cultural, political, operational, and physical. I draw upon several sources—my industry experience; the existing literature on organization design, Lean, and Agile; and several well-regarded works on individual and team psychology to present a synthesis for an Agile IT organization design. Many of the structural and operational configurations I suggest are already in place at several new-generation ISVs. I explain how the rest of enterprise IT could benefit from them. On the other hand, the chapters addressing politics and culture are equally relevant to ISVs and the rest of enterprise IT.

A number of reviewers suggested (in good faith) that I use Lean instead of Agile in the title in order to improve marketability. Apparently, Lean is “in” whereas Agile is jaded. However, I chose Agile because its strong people-orientation credentials are core to the solutions I offer.

Running counter to the trend of inflating an article-length topic into a book, I have packed a wide range of topics into a single book because I believe they are interlinked and a synthesis is required. It is also a sign of the wide scope of an IT leadership role. Yet the coverage isn’t exhaustive. Topics such as innovation, knowledge management, people diversity, and performance reviews are not covered or are only mentioned in passing.

Many of the topics covered here come under the scope of IT governance. But conventional IT governance gets into discussing standards and frameworks too soon. The discussion in this text is mostly standards and frameworks agnostic. Besides, the cultural aspects I address are not generally considered part of the scope of IT governance.

This isn’t a cookbook. I describe problems, explore causes, and offer solutions. However, I stop short of detailing steps to implement solutions. In a few sections, steps for migrating from existing situations are provided. In addition, the last chapter has a section that suggests a sequence of adoption of various recommendations. But overall, planning migration tends to be contextual. I expect that the intended audience will be able to draw upon the advice provided here and plan their own migration. Besides, I might be available for consulting.

Many of my recommendations here have succeeded in some shape in the real world. Wherever evidence of success or failure is publicly available, I have included it. In other cases, the recommendations are supported by reason and examples from first-hand experience. Once we accept that conventional approaches haven’t delivered needed results, it gets easier to consider the alternatives suggested seriously.

I provide many short, example scenario narratives to illustrate problems with the status quo. They are inspired by real situations, but names of companies, people, and other details have been altered. Any resemblance to real entities is coincidental and unintentional. They aren’t full stories in that they don’t include a resolution. However, the chapter content that follows the narrative provides ways to deal with the problem. The more you are able to relate the scenarios to your own experience, the more you will find the rest of the chapter to be illuminating. Many other inline examples are drawn from e-commerce (“e-tail”) so that they remain accessible to a diverse readership.

The first four chapters are a prerequisite to reading the rest of the book. Chapters 8, 9, and 10 on projects, finance, and staffing, respectively, go together. The rest of the chapters may be read independently. There are many cross-references between chapters despite my efforts to keep all the related arguments together. This is a sign that the topics are interdependent.

Given the broad scope, I had to bring up several subsidiary topics without going off on an explanatory tangent. To compensate, I have provided footnotes that link to freely available explanatory material from credible sources on
the Internet (rather than offline or behind a paywall). Readers of the physical version may access these links from the book’s companion website at agileorgdesign.com.

Who This Book Is For

- Execs and others who decide on matters of IT organization design or governance
- Senior management at ISVs and Internet businesses
- VP or director or head of IT, product management, engineering, or software development
- Business heads who interface with IT and IT-business partners
- Finance controllers, IT finance analysts, and investment managers
- Investors in digital businesses
- Techies who have the ear of executive leadership
- ICT strategists
- Members of IT governance groups
- Process quality/SEPG group members, process consultants, and coaches

This book is meant for medium to large IT organizations (50 to thousands of IT staff) that are facing challenges with IT and business agility. By IT organizations, I mean those that serve their own business directly (software is the business; e.g., ISV) or indirectly (Internet businesses, enterprise IT). That said, IT suppliers (IT services companies) might be able to use the contents herein to engage more effectively with their clients.

Why have I included investors in the target readership? Problems in organization design do not show up promptly in financial statements. Yet, in the long run, they have the potential to make or break business outcomes. Since investments tend to have a longer tenure than executives do, investors in digital businesses would do well to understand this topic and hold executives accountable.

If you have already achieved IT and business agility, the stuff here may seem obvious or old news. However, you are likely to encounter some new angles, arguments, or techniques. That said, this is not an introductory book by any means. It is assumed that the reader has some experience of software delivery and has at least a passing familiarity with Agile methods like Scrum or XP and the basic ideas of DevOps and continuous delivery.
This page intentionally left blank
Acknowledgments

Although I don’t have first-hand experience of caring for a newborn baby, writing this book did seem like how it might be. I could not imagine how my first draft of about twenty thousand words would grow into a healthy one-year-old of more than seventy thousand. And just like any IT project worth its overrun, I (and my editors at Addison-Wesley) have had to deal with scope creep and effort and schedule overruns. But staying true to the advice in this book, we tried to be value driven rather than plan driven. I believe that you will find value in it as well. A lot of it is due to the support, guidance, and encouragement of several people kind enough to help me. I am grateful to all of them.

Pramod Sadalage, a friend, colleague, and author of a number of books on applying Agile engineering techniques to data, was first to help. He recommended my work to Bernard Goodwin at Addison-Wesley, who then became my editor until his retirement at the end of 2014. Christopher Guzikowski then took over and brought this book to a successful launch. Michelle Housley provided great support throughout via developmental editing, coordinating with reviewers, and helping me get through paperwork. Stephanie Geels provided excellent copyediting, and Kesel Wilson and several others at Addison-Wesley ably helped with the production process. There are also others who I didn’t personally interact with, like the cover designer, compositor, and the proofreader. They have all been instrumental in bringing this work to fruition.

Pramod also reviewed my manuscript more than once. Early in this process, I was fortunate to receive a gracious offer of guidance from Jim Highsmith—a famous Agilist and also a colleague at ThoughtWorks. Thank you, Jim, for mentoring me over Skype calls and several e-mails. Dinesh Tantri who used to be digital strategist at ThoughtWorks helped crystallize my understanding of the digital transformation phenomenon. Ulrich John Solomon and Suresh Kumar Bellala from ThoughtWorks finance team helped me better understand the nuances of IT finance and validated some of my recommendations. Duncan Freke, development director at Trainline, enthusiastically supported my writing through several reviews and conversations around his own pioneering IT re-org efforts.

Special thanks to full-length reviewers who, despite their busy schedules, took time to read and offer detailed feedback: Aman King, Dave Farley, Dougal
Watt, Gil Broza, Keith Dodds, Randy R. Gore, Nagarjun Kandukuru, Sebastian Silva, Shyam Kurein, and Tom Poppendieck. Thanks also to other reviewers who managed to give the manuscript what time they could: Dave Whalley, Marco Abis, Puneet Kataria, Rajesh Babu, Rebecca Parsons, Sitaraman Dharmarajan, Sagar Paul, and Sunil Mundra. And of course, thanks to all those who read my manuscript and provided endorsements. Several colleagues helped obtain endorsements: Alagu Perumall, Anand Vishwanath, Vishwanath Nagraj Rao, and Sagar Paul.

ThoughtWorks and my office general managers provided invaluable support by granting me time to work on it. Jeremy Gordon offered friendly guidance on some legal aspects. Big thanks to Shabrin Sultana who is the creator of some of the better looking illustrations in this book. Thanks also to Siddharth Asokan from the ThoughtWorks marketing leadership team for making Shabrin available. The research team at ThoughtWorks also helped me with my research for the book.

I’d also like to thank Alanna Krause of Enspiral for permission to quote and use screenshots from her projects, Jo Freeman for permission to quote from her essay, and Henrik Kniberg for permission to quote from his writings and adapt one of his illustrations.

In a book on the evolution of buildings called How Buildings Learn, author Stewart Brand offers thanks to a “vast network of uncredited influence” that helped shape his work. I am similarly indebted to networks within ThoughtWorks and in the Lean and Agile community at large.
About the Author

Sriram Narayan, an IT management consultant with ThoughtWorks, has provided IT agility guidance to clients in telecom, financial services, energy, retail, and Internet businesses. He has also served as a leadership coach and a director of innovation. He was a founding member of the ThoughtWorks technology advisory board—the group that now authors Technology Radar. During a two-year stint at the products division of ThoughtWorks, he helped with product innovation and advocacy on Go—a tool that helps with continuous delivery. He has also worn the hats of a developer, open-source contributor, manager, product owner, tester, SOA architect, trainer, and Agile coach. An occasional blogger and speaker at conferences, his writings, talks, and contact information are available from sriramnarayan.com. The opinions in this book are his own.
This page intentionally left blank
Glossary

Note: The definitions here convey the sense in which a term is used in this book. They may not always be industry-standard terms or definitions.

Activity  Action that contributes to an outcome.

Activity-oriented team  A team that is responsible for a single activity. Usually a team of specialists (e.g., marketing, sales, support, development).

Agile  In the context of this book, the word Agile (capital A) is used to refer to a mindset or methodology that is aligned with the values and principles outlined in the Agile Manifesto. It is not used in the sense of the common English adjective.

Asynchronous communication  Channels of communication that don’t require parties to the communication to be available simultaneously. For example, e-mail and online forums are asynchronous while phone, VOIP, tele/web conference, chat, and face-to-face meetings are synchronous.

Build vs. buy or make vs. buy  A decision to build an IT solution (using in-house or outsourced talent) versus buy an off-the-shelf solution (increasingly in the form of SaaS).

Capability  In the context of this book, capability refers to the people and systems that make up a business-aligned IT capability.

CapEx  Expenditure of capital toward creating or enhancing assets (IT assets). It is recorded in the balance sheet. It shows up in the income statement only as an annual depreciation.

Continuous delivery (CD)  An approach to delivering software that reduces the cost, time, and risk of delivering incremental changes to users through seamless automation from development to deployment that makes production releases uneventful and frequent.

Continuous integration  A practice followed by Agile software development teams of frequently checking-in code under
development into a version control system, which then auto-triggers a comprehensive suite of fast-running tests after each check-in. It ensures that the codebase retains its functional integrity in the face of rapid development.

**Cross-functional team** An interdisciplinary, outcome-oriented team. It may consist of hard-core specialists, generalizing specialists, or generalists.

**Cycle time** The elapsed time for an item (feature) to progress through the complete value stream. Elapsed time = value-added time + wait time.

**DevOps** DevOps (development + operations) aims to improve collaboration between the development organization and IT-operations by locating these skills within a single team and by emphasizing culture, automation, measurement, and sharing.

**Digital business** A business that offers its customers a transaction space that seamlessly bridges digital and physical worlds.

**Digital transformation** Digital transformation is a change program that aims to transform a primarily brick-and-mortar business into a digital business.

**Function lead** A catch-all term in this book for people who provide leadership for specialist functions (e.g., VP or director or head of marketing, sales, development, architecture, quality, or program management).

**Handoff** The act of handing over a work item from one specialist or team to another. A value stream with a series of N specialist activities will have N – 1 handoffs.

**Internal scope** Scope internal to a feature. Flexible internal scope is key to leveraging a problem-solving approach as opposed to a deliver-to-planned-scope approach.

**Internet business** A business that doesn’t sell software but whose revenues are all (or mainly) via Internet transactions (contrast with ISV).

**ISV** Independent software vendor (increasingly of the SaaS variety). New-generation examples include companies such as Atlassian, Box.com, and GitHub.
IT-B  The part of the IT organization that creates value. The people in charge of conceiving solutions and building (and running) software. Wages of IT-B personnel are mostly treated as CapEx.

IT-I  The part of the IT organization that protects value. The people in charge of IT infrastructure and assets. Wages of IT-I personnel are mostly treated as OpEx.

OpEx  The ongoing, running cost of IT systems and infrastructure, including the wage cost of people dedicated to this. It shows up as expenditure in the income statement.

Outcome  An independently valuable and achievable business outcome.

Outcome owner  A catch-all term in this book for someone (below the rank of an exec) who is accountable for and dedicated to a business outcome. For example, product manager/owner/champion, chief product officer, or program/project manager.

Outcome-oriented team  A team that has autonomy and accountability for an outcome (e.g., a cross-functional product team).

SaaS  Software-as-a-service is a model of distributing software in which the vendor hosts the solution for the customer rather than it being installed on customer’s infrastructure.

Silo  Organizational silos are units or departments that tend to protect themselves and not work well with other units.

Systems of differentiation or engagement  The IT applications that help differentiate a business offering in the market or help drive engagement with customers.

Unscripted collaboration  Collaboration between teams is unscripted when it occurs outside of regular, scheduled meetings and without prior planning, permission, or approval.

UX (XD)  User experience (experience design).

Value stream  A value stream (in this book’s context) is a series of activities required to deliver a business outcome.
Chapter 5

Team Design

The first four chapters were short and introductory. The water gets deeper from here on. This chapter describes how various multiteam configurations, including the matrix organization, reduce organizational agility and how having fewer outcome-oriented, cross-functional teams can help. It explains why activity-oriented teams cannot work with small batch sizes required for lower cycle time. It covers why testing and maintenance are not separate activities and how certain configurations of outsourcing work better than others do. In terms of the key themes laid out in Chapter 3, the discussion in this chapter expands on organizing for responsiveness over cost-efficiency.

5.1 Framing the Problem

Why do business-IT organizations end up in situations where multiple teams are collectively responsible for a single business outcome? Here are some typical reasons:

- The scale of the problem is such that a single team would be unwieldy.
- Organizational boundaries
- Functional (activity-oriented teams)
- Regional (distributed teams)
- Business (product teams)
- Contractual (e.g., outsourcing)
- Shared support services (e.g., IT helpdesk, product support)
Whatever the reason for multiple teams serving a single outcome, once they are in place; it reduces the effectiveness with which the larger outcome may be realized. Why? Because collaboration within a team can be continuous, but collaboration between teams is always discontinuous (discrete). Meetings, for instance, are a great indicator of discontinuous collaboration. Continuous delivery needs continuous (and unscripted) collaboration. Effective collaboration on any given task between two individuals X and Y on two different teams depends on the following:

- Their individual dispositions to collaborate
- Their history of working together (relationship)
- The prevailing interteam communication protocol
- Whether the task has the same level of importance and urgency for both teams

The last two points can be affected by organization design. Can two individuals, X and Y, simply meet with each other, agree on what is required, and go back and do the work? Do they have to involve their respective managers in the process? Do the managers have to sanction the time for it? Does it all have to be via some system of record? The more process and indirection there is, the greater the friction for effective collaboration.

By contrast, people within a team don’t have to schedule meetings to collaborate with each other. They collaborate continuously and get into huddles (informal, ad hoc meetings—virtual or face to face) on demand. But given that multiple teams are unavoidable and that it reduces effectiveness, how can we design teams so that the most important outcomes are affected the least? This is the basis for the rest of our discussion in this chapter.

### 5.2 Activity-oriented Teams

Sales, marketing, product development, support, recruitment, and finance are all examples of specialized competencies. It is quite conventional to have a separate team per competency of this sort. Often called specialist teams, we call them activity-oriented teams to convey that they are formed around activities rather than outcomes (Section 4.1). Activity-oriented teams are a form of functional organization. In terms of traditional staff and line terminology, all staff and line functions are activity-oriented teams when they are organized separately by function.

For example, it is common to organize by specialization for a given line of products and assign a manager (full or part time) per line item below:

- Inside sales
- Field sales
- Sales engineers (pre sales)
- Marketing—content
- Marketing—advertising, social media
- Marketing—SEO, product web site
- Marketing—strategy
- Product management
- Product development
- Architecture
- UX
- Analysis
- Development
- QA
- Release management
- IT operations
- Product support
- Product solutions (custom installations, add-ons)
- Product training and certification

This effectively results in a dozen activity-oriented teams per product. Organizing teams like this isn’t the best way to serve the business outcome—that is, a successful product. It results in multiple, high-latency handoffs across teams to get anything done, whether it be developing a new feature, launching a marketing campaign for a product release, fixing a bug identified by a customer, or closing a new deal. Yet, it is what happens when IT-B is organized as a matrix.

### 5.2.1 Hamstrung by High-Latency Handoffs

As defined in Section 2.4.3, a value stream is a series of activities required to deliver an outcome. N activities require N – 1 handoffs for a work item (or batch) to pass through the value stream. Handoffs are simply a result of activity specialization. However, when a value stream is serviced by a series of
activity-oriented teams (functional organization), each handoff is a handoff between teams. This makes it slower and more expensive.

Consider the case where this work item is a software build. If the testing team is separate from the development team, they will not accept builds on a continuous basis but rather have their own schedule by which to take new builds. This means that each new build accepted by QA will have a lot more changes (large batch size) than in the case where new builds from development are automatically deployed into a QA environment on an ongoing basis.

Expensive handoffs encourage large batch sizes to reduce the total number of handoffs. A separate database team will not entertain piecemeal requests for query optimization. They’d rather own the data model and enforce indexing conventions across the board. They won’t review or help with unit-level database migration scripts. They’d rather review the whole set of migrations when the application is ready for UAT or some other similar state of maturity. On the other hand, a database specialist embedded in a development team will be much more responsive to piecemeal requests.

Large batch sizes lengthen cycle times. Items in the batch have to wait their turn for processing and, after processing, have to wait until all other items are processed before the batch can be handed over to the next stage. Even when all items are taken up for processing at once, the cycle time of the batch is at least equal to the cycle time of its slowest item. Long cycle times won’t do. There is mounting pressure to bring new capabilities to the market faster than ever.

In any system of work, the theoretical ideal is single-piece flow, which maximizes throughput and minimizes variance. You get there by continually reducing batch sizes.

—The Phoenix Project

Short cycles require small batch sizes. Reinertsen argues that reducing batch size helps reduce cycle time, prevent scope creep, reduce risk, and increase team motivation. Reducing batch size is impractical when handoffs are expensive. Recall that a value stream with N activities requires N – 1 handoffs per batch. Halving batch size doubles the total number of handoffs needed. This is only feasible when handoffs are inexpensive; that is, when we move away from using multiple activity-oriented teams to service a value stream. Figure 5-1 summarizes the discussion thus far in this section.

2. (Kim, Behr, and Spafford 2013)
3. (Reinertsen 2009)
Figure 5-1 Team design influences batch size.
5.2.2 The Traditional Lure of Functional Organization

Why has functional organization persisted over the years despite the drawbacks described above? The traditional motivation for specialized teams can be traced to a legitimate desire for:

- Efficient utilization of specialist resources across a line of products: Rather than dedicate, say, two specialists to each of four products with an average specialist utilization of say 60%, it is more efficient to create a shared activity-oriented team of five (since \(2 \times 4 \times 0.6 = 4.8\)) people available on demand to any of the four products. This is also an attractive option in a situation where supply of the said specialty in the market is scarce.

- Standardization: As members of a single specialty team, say, a marketing content team, it is easier to standardize templates and formats, achieve consistent messaging across product lines, and coordinate product releases.

- Nurturing the competency by localizing it: When people of a common specialization sit together, it is easier to share knowledge and help each other with troubleshooting, think through a solution, review each other’s work, etc. It is also easier for the team manager to ask for a training budget and other resources.

The traditional model has come under question because of the increasingly shorter time to market and time in market.\(^4\) Software products have a very short window available to monetize new features or capabilities. We can no longer take for granted an entrenched customer base; it is likely their patience will wear out unless they see a steady delivery of valuable capability. Even in the case of enterprise IT, being responsive to the business is more important than minimizing cost per function (or story) point. The traditional model of activity-oriented teams may be good for cost-efficiency, but it is bad for end-to-end cycle time. It is therefore worthwhile to trade off some efficiency for the sake of responsiveness. As we will see in Section 5.4, a cross-functional team is a good way to achieve this tradeoff.

Just enough standardization and consistency can still be achieved without being part of the same team. It is harder but possible, as we will see later from the Spotify example. On the other hand, specialist teams have a tendency to adhere to a mindless uniformity across all sorts of unnecessary things in the name of consistency across the product line.

\(^4\) http://www.thoughtworks.com/insights/blog/rise-serial-innovator
As for nurturing competencies, it is important, but not at the expense of the business outcome. Organization design ought to cater to first things first. There are other ways of nurturing competencies like cultivating communities of practice. More on this in Section 5.7.

### 5.2.3 When Is It OK to Have Activity-oriented Teams?

What about departments like HR, admin, legal, and finance? Are they organized around outcomes or activities? If we go by how we distinguish between outcomes and activities in Section 4.1, it is clear that these support functions don’t own independently valuable business outcomes. Therefore, they are activity-oriented teams. Does it then mean they automatically become silos and therefore candidates for being disbanded?

Some activities are closer to the outcome than others. For example, UX is closer than admin to the outcome of product success. Ask whether the realization of the outcome is dependent on repeated successful iterations through some core value stream. If yes, then the activities belonging to this value stream should not be conducted in separate activity-oriented teams. Activities that aren’t an integral part of a business outcome’s core value stream may be spun off into separate teams without much risk.

Even where they are not part of a value stream, activity-oriented teams tend to standardize their operations over time. Their appetite for offering custom solutions begins to diminish. Complaints begin to surface—“They threw the rule book at us,” “What bureaucracy!” and so on. However, as long as they don’t directly affect business outcomes, they are allowed to exist.

For example, it is an anti-pattern to maintain a long-lived knowledge management (KM) team. It is an activity-oriented team for what is meant to be a collective activity. Disband it after initial rollout of the KM system. KM is everyone’s responsibility. Knowledge is documented via recorded conversations, videos, blog posts, proposals, and reports. Let the relevant community of practice (Section 5.7) curate its content on the KM system. It is generally so specialized that it doesn’t help to hire a generalist technical writer or content curator.

### 5.2.4 Independent Testing, Verification, and Validation

Independent testing is the notion that the team that tests should be different and separate from the team that develops in order to achieve greater rigor in testing. Many IT services vendors offer independent testing services. Doesn’t this justify a separate activity-oriented team for testing? In my experience, there is no loss of rigor or conflict of interest in including developers and testers on the same team. Any deficiency in testing is bound to show up in UAT or production and
reflect poorly on the team or the vendor. Given the cost of acquiring new clients, IT suppliers are generally extremely keen to land and expand, that is, cultivate long-term relationships and grow accounts.

On the contrary, independent testing wrecks the flow of work through the development value stream. It discourages collaboration between developers and testers and leads to all sorts of suboptimization by both teams to protect their reputations. The chapter on metrics (Chapter 12) describes a number of scenarios of suboptimization resulting from independent testing.

Hiving off testing for lack of in-house skills is a different matter altogether. For example, it is common to engage a third party for testing security—vulnerability assessments, penetration testing, etc. However, this doesn’t come in the way of the development value stream as much because it is somewhat removed from the functionality being built.

Then there are those who argue that verification and validation activity should be conducted at arm’s length from each other. But the traditional distinction between software verification and validation is old school. One distinction is that validation is akin to field tests while verification is closer to lab tests. In case of pure software, A/B tests and beta customer programs come close to field tests whereas tests of functionality and simulated performance tests are closer to lab tests. Although the distinction makes sense, it is no reason to separate the people that perform field and lab tests from each other and from the rest of the development team. A second oft-quoted distinction also makes sense in this light but is rarely applied correctly. It is said that verification checks whether we have built the thing right, and validation checks whether we have built the right thing. However, in practice, we frequently find no provision for field tests and so-called validation teams are responsible only for end-to-end lab tests, while verification teams are limited to component-level lab tests.

### 5.3 Shared Services

Shared services are similar to activity-oriented teams except that they are usually shared across unrelated business outcomes. All shared services are activity-oriented teams, but all activity-oriented teams aren’t shared services. For example, if a product development team is split into a team of developers and a team of testers with a manager per team, they are activity-oriented teams but not shared services. Typical examples of shared services include IT helpdesk,

---

software product level-2 support (single team serving multiple products), internal private cloud team, and call centers. Although they play a crucial supporting role in the realization of business outcomes, they are often treated and managed purely as cost centers. Shared services cannot be totally avoided, but they shouldn’t be encouraged as a way to do more with less. It is usually counterproductive to have enterprise architecture, UX, software testing, IT operations (e.g., for a SaaS product) or even product marketing and sales as shared services. Ethar Alali has written a great two-part article explaining the drawbacks of shared services and activity-oriented teams with a non-IT example.

5.3.1 Shared Services Lose Purpose

When several teams of developers share a common team of testers, what is the purpose with which the testers identify? The developer teams each have a product to develop; their purpose is a successful product or at least a successful release. The shared testing team’s purpose often degenerates to that of being an efficient provider of testing services with allegiance to no particular product.

It is important to recognize this aspect of shared services. By definition, shared services are used by teams responsible for different business outcomes. The shared team itself isn’t responsible for those outcomes. It is no surprise then that we sometimes get the feeling of dealing with mercenaries when interacting with a shared service team. They don’t seem to have their skin in the game.

Shared services struggle to find purpose. An organization design that aims for conditions of autonomy, mastery, and purpose should strive to minimize shared services and eliminate them from mission-critical value streams.

5.3.2 Reducing Friction in Shared Service Interfaces

Interteam collaboration typically requires following a communication protocol enforced by a work tracking tool or a single point of contact. It means meetings between team representatives with documented minutes of meetings. Feedback loops lengthen, reducing our ability to fail-fast (Section 2.4.1). Team managers try to showcase their team’s performance with team-level metrics. Incoming work gets queued and prioritized based on some centrally conceived criteria. Dependent teams get frustrated with turnaround times and attempt priority escalations.

Here is an example of how a communication protocol designed for cost-efficiency ends up affecting responsiveness. It is typical for IT support to use ticketing systems. It helps the IT support manager track the workload. Some

employees who are friends with the engineers in IT support tend to request help directly via chat. This is understandably discouraged because it flies under the manager’s radar and does not leave an audit trail. Once a ticket is assigned to an engineer, she is expected to carry out the task and put the ticket in the state of “completed subject to customer confirmation.” Sometimes, the ticket lacks some information or it needs troubleshooting on the requestor’s computer. Depending on the nature of the ticket, she may choose to:

- Reply to the ticket asking for more information and put the ticket in a state of “awaiting customer response” or
- Get in touch with the requester via phone/chat, obtain the needed information, carry out the task, and close the ticket.

The first option is probably more efficient from an IT support perspective. She doesn’t have to look up the requestor’s phone number and there is no wasted communication effort if the requestor is not available at that moment. Besides, everything is written down and recorded. The second option is more responsive from the requestor’s perspective. It feels less like dealing with a bureaucracy.

The first option can get worse for the requestor if the ticket is reassigned to a different engineer every time the requestor responds. We experience this first-hand when trying to sort out a nontrivial problem with our bank or telecom provider’s call center. We are expected to explain the whole problem all over again to a new agent. Being able to switch agents freely on a ticket helps maximize agent utilization. Unfortunately, it also maximizes customer frustration.

Designers of the system may argue that the history of the ticket is recorded, and so the customer should not have to repeat it. However, the recorded history is seldom self-explanatory. Besides, an agent new to a ticket would much rather hear it again first-hand than having to read through and assimilate the record.

What if the situation is level-3 commercial product support for external customers? Getting in touch with the requestor might be unrealistic, but we could at least have the same person responding until the ticket is resolved. What if, in order to provide 24×7 support, level-3 people are located in different time zones? Now we can’t help agent switching, can we? Well, at least we can avoid agent switching within a time zone on a given ticket.

5.4 Cross-functional Teams

A cross-functional team (also called multifunctional, poly-skilled, or interdisciplinary) is one whose members belong to different specializations and work together toward a common outcome. They are a necessary consequence of
organizing for business outcomes rather than activities. The realization of any outcome is bound to involve many different activities. This calls for people with widely different skills to be part of the same team. For example, a cross-functional product team may consist of people with all the skills listed in Section 5.2.

The top half of Figure 5-2 shows a conventional stratified IT organization. The product owner is quite removed from daily development. The term development team is only applied to a minimally cross-functional team of developers, testers, database, and UX people. Sometimes it is worse—development team just refers to developers. In either case, the team is not equipped to own a business outcome.

The lower half of the figure depicts what it would take to own an outcome. The inner box represents a well-equipped cross-functional product development team. Architects, business analysts, deployment engineers, and product owners join the team. Some parts of IT operations, marketing, and sales are also folded in. For example, Operations-A provide a virtualized platform that Operations-B uses for test and production deployments. Field sales and inside sales (Sales-A) may sit outside, but sales engineers (pre-sales) could very well be part of the team. Similarly advertising, SEO, promotions, and pricing (Marketing-A) may sit outside, but social media and content (Marketing-B) would do well to be part of the team.

Cross-functional teams aren’t a new idea. Only the proposed extent of cross-functionality is new. Agile software development teams have always been

![Figure 5-2 Moving from a stratified setup to a cross-functional setup](image-url)
cross-functional with respect to architects, analysts, developers, and testers. With DevOps, cross-functionality expands to include deployment and some IT operations people. At this point, the cross-functional team is capable of agility in delivery. For full IT and business agility, the circle needs to expand further to include dedicated product owners, UX people, sales, marketing, and support.

5.4.1 DevOps = Cross-functional Dev + IT Ops Team

The DevOps movement advocates a merger of development and related IT operations. It makes a team cross-functional with respect to development and IT operations. Unfortunately, this aspect is often ignored in comparison with the technical aspects of DevOps. From an IT point of view, we broadly have three departments in a typical setup—business, development, and IT operations. There may be many more subdepartments, but this picture is enough to understand what DevOps is not.

In a typical case, once the VP of IT operations is convinced about DevOps, she decides that her team should now acquire so called DevOps capability. Accordingly, they evaluate and buy some product claiming to be a DevOps enabler, do a bit of research on virtualization and infrastructure automation tools, start version controlling their deployment scripts, and then rename their department to DevOps. Is it really DevOps? Well, it isn’t DevOps if you don’t have IT operations people as part of your development organization. The whole point of DevOps is to locate development and operations skills within a single team. The VP of IT operations is not to blame though. Effecting a DevOps reorg is usually beyond her pay grade and fraught with implications for her future role.

5.4.2 Organizing for Responsiveness

It Works!
@Apple
Apple uses cross-functional teams as part of its Apple New Product process (ANP). Cross-functional teams are used for product discovery and definition, product development, and even to define the ANP.8

---

8. http://www.roundtable.com/download/db8e1aaf0cb3aca1ae2d0018624204529/9778d5d219c5080b9a6a17bef029331c
Cross-functional teams fold the entire software delivery value stream into a single team rather than let it span across multiple activity-oriented teams. This reduces the cost of handoffs, allows reduction in batch size, and thereby decreases cycle time (improving responsiveness). Cross-functional teams aligned to outcomes can get meaningful things done within the bounds of the team. In this respect, they are much more autonomous units than activity-oriented teams. They are also more fun to be since autonomy is an intrinsic motivator.

Cross-functional teams aren’t anti-specialization. These teams still consist of specialists. Specialization isn’t the problem; organizing along the lines of specialization is. Functional organization makes for slower and more expensive handoffs.

---

5.4.3 Utilization

Will specialists dedicated to product teams be underutilized? Probably yes. This is where the rubber meets the road in terms of trading off cost-efficiency for the sake of responsiveness. Beyond a certain threshold of utilization, responsiveness decreases as utilization increases. This is a well-known effect from queuing theory. Without some slack, we can’t have responsiveness. A fully utilized highway is a parking lot.\(^{11}\)

Besides, as a side effect of being part of a cross-functional team, specialists usually start to acquire adjacent skills. So developers pick up infrastructure skills while product analysts pick up testing skills. This lets them contribute to adjacent areas during lean intervals and jump back to core areas as soon as something comes up. Pure specialists start morphing into generalizing specialists.\(^{12}\) Their skill profile changes shape from the letter I (all depth) to the letter T (some breadth).

5.4.4 T-shaped People

Pure specialists are all depth and very little breadth. Although they may be brought together in a cross-functional team, they might face challenges in interacting with their team members from other specializations. For effective cross-functional collaboration, we need some breadth as well as good depth. Breadth provides perspective and empathy. Hard-core specialists are susceptible to caring only about their part of the work. T-shaped people\(^{13}\) can relate to and build upon ideas coming from outside their domain with greater ease.

5.4.5 Team Size

Common recommendations for development team size range from three to nine.\(^{14,15}\) Another idea called the two-pizza team (number of people that two pizzas will suffice for) comes from Amazon. As long as the architecture is modular (via services or otherwise), these are reasonable heuristics for team size of a single module or service. However, highly cross-functional, outcome-oriented teams, as in Figure 5-2, are likely to be much bigger if the outcome (or suboutcome) requires it. It doesn’t mean humungous standup meetings or that everyone communicates regularly with each other. The cause of responsiveness is

\(^{11}\) Tweet by Paul Sutton, @FragileAgile.
\(^{12}\) http://www.agilemodeling.com/essays/generalizingSpecialists.htm
\(^{13}\) http://en.wikipedia.org/wiki/T-shaped_skills
\(^{14}\) https://www.scrum.org/Forums/aft/680
served by having a single, dedicated outcome owner for the whole team of teams. The cause of autonomy and purpose is served by having a team big and capable enough to own a business outcome (or suboutcome).

5.5 Cross-functionality in Other Domains

The notion of cross-functional organization is also pertinent to disciplines other than IT. It is only fair to indulge in a few interdisciplinary analogies while on the topic of interdisciplinary (cross-functional) teams.

5.5.1 Hospital Pod Teams

A study\(^{16}\) conducted at the emergency department (ED) of a city hospital corroborates the advantages of moving from an activity-oriented team design to a cross-functional one. Their initial design had three activity-oriented teams of nurses, residents, and attending physicians servicing a value stream that consisted of the following activities:

- Triage incoming patients
- Begin patient care work (nurse)
- Order tests, make decisions about diagnosis, treatment, and disposition (resident)
- Approve or change the orders and decisions (attending physician)

The study notes that back-and-forth discussion was not enabled by this design. Responsiveness was poor—an average of 10% of patients left without being seen because of delays. As a redesign, the ED teams were divided into pods (cross-functional teams). Each pod had the personnel and equipment necessary to treat any type of ED patient; that is, it had the ability to service the entire value stream above. The study found that the pod system delivered a 40% reduction in cycle time (in this case, cycle time is the average amount of time a patient spends in ED) without any significant difference in any other aspect of the quality of care. Note that the rest of the hospital functions can continue with an activity-oriented organization, as they are not directly part of the patient-care value stream.

---

5.5.2 A Cross-functional Museum Layout

The Rijksmuseum in Amsterdam is a good example of the power of cross-functional organization. Traditionally, museum galleries have a functional organization—one gallery for sculpture, another for ceramics, a third one for paintings, and so on. Each gallery is managed by a specialist curator—the museum analogue of our function lead. But the new Rijksmuseum has opted for a more integrated or, shall we say, cross-functional organization. Each section in now devoted to a different century, and within that section you will find all the artifacts from that period arranged in an integrated holistic display that effectively conveys the story of the age.

An article about the reopening of the museum in *The Guardian*\(^1\) describes the new layout. A Rembrandt gallery, for example, displays some of his early work alongside period-piece furniture, glass and silver artifacts made by people he knew, and a portrait by an art-patron friend. Rijksmuseum’s director of collections, Taco Dibbits, says, “You get a sense of the world in which Rembrandt was producing his art.” This is similar to how a product analyst gets a sense of the world into which the product is deployed by working in a cross-functional team that includes deployment specialists.

A cross-functional layout is arguably more work for the curators to manage and maintain. It may also nettle expert visitors who may be interested, for example, in sculpture but not ceramics. But from the point of view of majority generalist visitors to the museum (the outcome that matters), a cross-functional layout is probably more meaningful.

5.5.3 Taskonomy

Design guru Dan Norman talks about taxonomy versus taskonomy in the context of human-centered design.\(^2\) Imagine how much less usable a word processor or spreadsheet might be if it only supported main menus, that is, no support for context-sensitive (right-click/pop-up menu) actions. As exemplified by Figure 5-3, the main header menu of an application is taxonomy whereas its myriad context-sensitive menus are taskonomies. Taxonomy is a functional classification or arrangement whereas a taskonomy is a cross-functional arrangement based on the needs of the task at hand. Taxonomies provide navigability—they offer a map of available functionality. Taskonomies provide ease of use and responsiveness—they are responsive to the needs of the user in context. In a user interface, both have their place. In organization design, the

---

1. http://www.theguardian.com/culture/2013/apr/05/rijksmuseum-reopens-long-refurbishment-rethink
org chart provides enough taxonomy. For responsive day-to-day work, we need
taskonomies, which is what cross-functional teams are.

5.6 Migrating to Cross-functional Teams

It is quite disruptive to move from an IT-B matrix organization (or other func-
tional organization) to self-sufficient, cross-functional teams. Here is one
method of doing it gradually:

1. Identify products/capabilities that differentiate the business. You will need
   as many cross-functional teams as the number of differentiating business
   products/capabilities.

Figure 5-3 Taxonomy is to taskonomy as functional arrangement is to cross-functional
arrangement.
2. Identify a product/capability for piloting the transition. Ideally, the candidate won’t have too many dependencies with other products/capabilities. Make sure there is a full-time outcome owner (Section 4.1.2) available.

3. Have the product owner come up with an initial product roadmap and backlog.

4. Identify people from existing activity teams that will make up the pilot team. Explain to them the rationale for the pilot. Use the penny game\textsuperscript{19} to drive home how small batches and inexpensive handoffs help reduce cycle time.

5. Make sure the pilot team has all the skills required to be self-sufficient.

6. Let the new team start working through the backlog.

7. See how it goes for about three months before deciding to spin up another cross-functional team.

This only addresses the structural aspects of migration. Operational, cultural, and political aspects are addressed in the following chapters.

### 5.6.1 Separation of Duties

Sometimes, IT governance people say that cross-functional teams are not permitted by accounting and investor protection regulations such as SOX and payment regulations such as Payment Card Industry Data Security Standard (PCI-DSS). In particular, they speak of a control called \textit{separation of duties}.\textsuperscript{20} In effect, it aims to separate authorization for making changes to an application/system from authorization to release those changes into production. Traditionally, this hasn’t been a problem because the production deployment team was different from the development team. However, even if separation of duties requires that the same person not have both authorizations, it does not prohibit two people with the combination of authorizations from working together on the same team.\textsuperscript{21}

---

\textsuperscript{19} http://www.leansimulations.org/2014/04/ Variations-of-Lean-Penny-Game.html

\textsuperscript{20} http://en.wikipedia.org/wiki/Separation_of_duties

\textsuperscript{21} http://continuousdelivery.com/2012/07/ PCI-DSS-and-Continuous-Deployment-at-Etsy/
5.7 Communities of Practice

We saw earlier that a cross-functional team encourages its members to morph from pure specialists to generalizing specialists. This does not have to come at the cost of mastery in their specialization. A community of practice (CoP) is an alternative solution to nurturing a competency in the absence of a functional organization. A CoP does not require its members to be all part of the same team. It functions like a loose, professional association of specialists with mechanisms for online and offline interaction and knowledge sharing.

A lead is usually elected, nominated, or appointed per CoP. The lead comes from the same specialist background and is someone with people and organizing skills. The CoP lead is by no means a full-time role—she continues to work as a first-class member of some product team while devoting maybe 20% of her time to CoP work. CoP leads sponsor brown bag sessions, training programs, internal conferences, and sponsor members to participate in external conferences. They weigh in on tools and modes of collaboration within the community. They are accountable for the health of the community.

In addition, mastery in IT specialist areas may be sustained by getting involved in groups and activities outside one’s organization. Specialist user groups and conferences are thriving in many cities. The Internet has many great resources for specialist skill enhancement. Even just following relevant Twitter hashtags goes a long way toward staying up to date. After all, individual mastery is at least as much the individual’s responsibility as the organization’s.

5.8 Maintenance Teams

Cross-functional product teams own their product—they shape it, build it, maintain it, and run it. However, many organizations retain separate teams for maintenance (bug fixes and minor enhancements) and IT operations.

Figure 5-4 shows a traditional cycle. Maintenance and IT operations work on what is released while development works on the next release. To cater to users who cannot upgrade to newer releases promptly, there is usually a support window of current minus N releases \(N = 1\) in Figure 5-4.

There is common but flawed notion in enterprise IT circles that maintenance work requires less skill than full-scale development. As a result, project sponsors looking to reduce cost opt for a different team of lower-cost people for maintenance work. This is false economy. It hurts the larger business outcome
and reduces IT agility. When the same product team does development and maintenance, there is no handoff at release time. It is easier to merge bug fixes from release branch to trunk because the team is familiar with the ongoing changes in trunk. What’s more, trunk-based development—a branchless technique that is gaining adoption—is nearly impossible with separate development and maintenance teams.

End-of-life support is one situation where a maintenance team might make sense. This team keeps an old app/product running while a new replacement is built. Other than that, it is all about tearing down potential silos such as separate maintenance teams. Even in case of end-of-life support, a capability-oriented IT organization may choose to have the old and new coexist in a single capability team (Section 8.2). A separate maintenance team is a dinosaur in an age of continuous delivery and DevOps.

5.9 Outsourcing

When IT-B work is outsourced, we need to take care that the resulting team design does not violate the conditions of responsiveness, autonomy, mastery, and purpose discussed previously. Otherwise, business outcomes may be at risk. For example, the CapEx-OpEx distinction results in separate contracts/teams/vendors for development and maintenance. Some organizations go a step further and outsource even IT operations to a different team/vendor under a separate contract. The rationale is to stick to core business competency and outsource everything else (let vendors compete with each other for our slice of IT). Depending on how critical an application is for revenue generation, this strategy of “divide-and-conquer IT” can be frustrating at best and suicidal at worst. Internet businesses and ISVs typically outsource little to none of their IT-B. This is simply because having to orchestrate between three teams/vendors for every new feature is a huge drag on the ability to go to market quickly.

---

22. http://paulhammant.com/2013/04/05/what-is-trunk-based-development/
Equally important, the feedback loop is badly constricted at contractual boundaries. Designing formal, service-level agreement (SLA)-driven protocols of communication between business, development, IT operations, and maintenance is a recipe for bureaucracy and indifference.

Outsourcing along outcomes is better than outsourcing along activity lines—that is, consider outsourcing application A to vendor X, B to vendor Y, and C to vendor Z (Figure 5-6) rather than handing development of A, B, and C to vendor X, IT operations to vendor Y, and maintenance to vendor Z (Figure 5-5).
Outsourcing outcomes (or suboutcomes) is the first step. The next step is to ensure that the vendor follows the same practice while internally organizing for delivery of the outcome. Many vendors adopt a utilization-friendly, activity-oriented organization internally, thus defeating the intent of the outsourcing configuration.

On the other hand, it may be that your business doesn’t change that often or your application isn’t strategic. If so, it is useful to ask, “Why make (build)? Why not buy?” SaaS is mainstream. It is likely that someone is offering your bespoke application as a service. At the cost of some tweaks to your business process and a one-time migration, you might end up with a better application at lower cost. The SaaS vendor in turn is likely running a fully in-house IT-B setup.

5.10 The Matrix: Solve It or Dissolve It

Half the world is so used to matrix management as to take the scheme for granted.

The other half just thinks it’s bizarre.

—Tom DeMarco in *Slack*, p. 15

A matrix structure is one whose members have two bosses—typically a project manager for day-to-day work and a longer-term function lead for performance appraisals and training. In case of IT-B, the project managers work with product owners who either come from the business or liaise with people from the business. Function leads in IT-B have titles like head/VP/director of architecture, development, UX, database, testing, or release management. Function leads own “resources” (e.g., developers, testers) who get assigned to projects as needed. Given that IT is itself a “function,” an IT-matrix represents a functional organization within a functional organization—a near guarantee of pain. From business’s point of view, they are the verticals (lines of business) in the matrix and the different IT functions are horizontal. From IT-B’s point of view, the functions are verticals and the different projects are horizontals. As shown in Figure 5-7, we use the latter frame for our discussion. The verticals in an IT-B matrix are activity oriented whereas the horizontals (projects) are outcome oriented. As work moves through the software delivery value stream, it is handed over from one vertical to the other in the IT-B matrix. As explained in Section 5.2.1, these handoffs present a structural impediment for short cycle times.

Matrix structures are probably okay where the verticals don’t have to engage with each other in a fast-moving value stream; for example, a sales organization.
may be set up as a matrix with verticals for different product lines and horizontals for different regions. However, a matrix is inappropriate for an IT-B organization that aims for continuous delivery. Continuous delivery requires continuous collaboration—a lot of it unscripted. It is something with which the verticals in a matrix simply can’t cope.

While no matrix structure is suitable for continuous delivery, some are worse than others. In the following section, we’ll explore different types of matrices and contrast them with cross-functional teams.

A handoff between two verticals in a matrix can be represented as a queue; for example, development does its work and puts it in the queue of the testing team. A vertical may have a single queue for all incoming work or one queue per project. In the latter case, specific people may be assigned to handle a given project’s queue or it might just be a capacity allocation without fixed people assignment. The relative merits of various configurations are illustrated in Figure 5-8 and discussed below. Modern business by necessity trades cost-efficiency for responsiveness because business agility is a critical success factor.

5.10.1 Matrix of Shared Services

A matrix of shared services allocates both capacity and people just in time i.e. all projects share a single queue for a given function. There is no certainty of available capacity for projects. Wait times are indefinite but resource utilization is maximal. This is the worst possible matrix configuration for continuous delivery.
5.10.2 Matrix with Dedicated Capacity and Fungible People

In this case, every project gets its own queue and a certain number of full-time equivalents (FTEs) to service the queue, but the actual people who make up the FTEs aren’t fixed. Although this makes for flexible work assignment, there is drastic loss of context as people switch tasks.

5.10.3 Matrix with Dedicated Capacity and People

Here, we assign a fixed set of people to a product for an agreed-upon period of time. Product owners still have a tough time getting their work done through the different layers from left to right. Occasional power struggles break out between outcome owners and function leads. It is still bad in the sense that there are too many handoffs. There is a tendency for batch size to go up. It does not encourage continuous collaboration, and hence, we see a lot of meetings taking place.

In my experience, a matrix organization can achieve monthly releases at best. But release interval is not the same as cycle time. Monthly releases imply a minimum cycle time of a month, very likely much higher; for example, it might take six months for a new feature to move through all the verticals of a matrix before it is released.
5.10 The Matrix: Solve It or Dissolve It

5.10.4 Monolithic Cross-functional Product Team

Figure 5-9 shows self-sufficient cross-functional product teams. The product team is fully accountable for the success of the product. It is almost like a different business unit except that they still depend on external shared verticals such as finance, admin, legal, and HR. Each product team has one person in charge as the outcome owner.

5.10.5 Cross-functional Setup with Activity-oriented Subteams

A single monolithic team may be unworkable after a certain size. At that point, the outcome owner may choose to assign an additional manager to the largest groups of specialists, for example, a manager for the developers or the inside salespeople.

5.10.6 Cross-functional Setup with Outcome-oriented Subteams

It is better to scale big product teams by creating teams that own suboutcomes rather than activities. Apart from the advantage of responsiveness, this also

![Monolithic cross-functional product teams](image-url)
promotes modular software architecture. Conway’s law\textsuperscript{24} states that the design of a system is likely to reflect the communication structure of its team. Accordingly, monolithic teams tend toward monolithic architectures, layered teams (separate teams for front end, business logic, persistence, etc.) tend toward layered architectures, and teams that own different product modules will tend toward a modular architecture. The ThoughtWorks Technology Radar called it the “Inverse Conway Maneuver.”\textsuperscript{25}

### 5.11 Summary of Insights

- Collaboration within teams tends to be unscripted—on demand, just in time, and continuous. Collaboration across teams tends to be discontinuous and discrete (e.g., via meetings). This can befactored into team design by locating all roles that require continuous collaboration within a single team.

- Handoffs are mostly a result of specialization. Organization design cannot reduce these handoffs, but it can make them faster and cheaper by making them occur inside a single team.

- The biggest promise of continuous delivery is a reduction in IT delivery cycle time. It requires the delivery value stream to process work in small batches, which ultimately calls for a single team (or as few as possible) responsible for the whole value stream.

- A cross-functional team consists of people with different primary skills working toward a common goal. They are an example of valuing responsiveness over cost-efficiency.

- Cross-functional teams aren’t anti-specialization. Specialization isn’t the problem; organizing along lines of specialization is.

- It is okay to have activity-oriented teams for activities that aren’t an integral part of a business outcome’s core value stream.

- Release interval is not the same as cycle time. Monthly releases imply a minimum cycle time of a month, very likely much higher.

- Given that IT is itself a “function,” an IT-matrix represents a functional organization within a functional organization—a near guarantee of pain.

\textsuperscript{24} http://www.thoughtworks.com/insights/blog/demystifying-conways-law

\textsuperscript{25} http://www.thoughtworks.com/radar/techniques/inverse-conway-maneuver
5.12 Summary of Actions

- Shift from activity-oriented organization to outcome-oriented cross-functional teams. They tend to be self-sufficient (autonomy) and business goal-directed (purpose).

- Communities of practice complement outcome-oriented organization. In the absence of a functional organization, they provide the necessary umbrella to nurture specialist competencies.

- If needed, divide work among multiple teams by splitting the outcome into suboutcomes (e.g., modules of an application or different applications) rather than activities (development, testing, etc.). The same applies for geographic distribution and outsourcing.

- Don’t encourage creation of shared services that service critical value streams. They tend to lose the sense of business purpose and this hurts responsiveness.

- Don’t commission separate teams for maintenance. It is an example of unnecessary handoff created by team design. Besides, rarely is anything in a pure maintenance mode. It always coexists with forward-looking development. A separate maintenance team is a dinosaur in an age of continuous delivery and DevOps.

- Move away from an IT-B matrix to outcome-oriented teams aligned with business verticals.
Index

ABC (activity-based costing), 127
Ability, disadvantages of separating planning from execution and, 93
Accelerate (Kotter), 10
Access control
  freedom and, 184
  on need-to-restrict basis, 149–150, 230
  organizing for responsiveness over cost-efficiency, 229
  silos related to tool access, 151–152
Accountability
  in absolute hierarchies, 83
  assigning, 78
  balancing autonomy with, 77
  decision making and, 86–87, 229
  decision record and, 88–89
  designing for intrinsic motivation, 230
  disadvantages of separating planning from execution, 93–95
  mapping. See Accountability maps
  matrix paralysis and, 82–83
  migrating from matrix to professor-entrepreneur model, 86
  minimizing power struggles, 82
  opposition in planning process and, 96–97
  org chart debt, 97
  outcome ownership, 78–79, 85–86
  overlapping planning and executing, 96
overview of, 75
planning and executing, 92–93
power and hierarchy and, 75–77
professor and entrepreneur model (Poppendieck), 83–85
resistance to decision record system, 91–92
scope of decision record system, 91
summary, 98
tools for, 89
Accountability maps
  clarifying accountability, 230
  decision ownership and, 89
  as information radiator, 98, 234
overview of, 79–82
Accounting. See also Budgeting
  classification of activities, 129–130
  relevance of, 125–126
  team design and, 228
  without timesheets, 128–129
Achievable
  qualities of good outcomes, 38
  qualities of good tasks, 38
  testing sub-outcomes, 46
Activities
  avoiding activity-oriented outsourcing, 67
  classifying in accounting, 129–130
  defined, xxvii
  in execution phase, 92–93
  forming teams around outcomes not activities, 46
  handoffs resulting from specialization of, 49–51
Activities (continued)

outcome-oriented metrics preferred over activity-oriented metrics, 176
outcomes based on chain of, 37–39
serving outcomes, 45
Activity-based costing (ABC), 127
Activity-oriented teams
appropriates uses of, 53, 72
compared with outcome-oriented teams, 43
defined, xxvii
functional organizations and, 52–53
in GICs, 242
high-latency handoffs impacting, 49–51
independent testing, verification, and validation in, 53–54
overview of, 48–49
purpose diminished by, 231
shared services as form of, 54–55
shifting to outcome-oriented teams, 73
specialization in, 39–40
spurious implementations of Agile development and, 22
Adaptability
adaptability metrics preferred over predictability metrics, 176–177
metrics and, 228
vs. predictability, 161
Adobe Kickbox, 215
Aesthetics, meaning trumping in visual aids, 213–214
Aggregate metrics, preferred over fine-grained metrics, 176
Aggression. See Microaggression
Agile Manifesto
the Agile credo, 13
Agile culture, 16–17
changes in Agile development over time, 1
continuous delivery and DevOps and, 15–16
disillusionment with Agile development, 21
examples, 14–15
fail-fast principle, 18
information radiators, 20–21
iterative development preferred over incremental development,
18–20
spurious implementations of Agile development, 22
statement of, 14
summary, 22
value stream optimization, 20
Agility
Agile defined, xxvii
organizational agility touching all aspects of business, 7
problems/solutions in organizational agility, 228
translating team-level agility into IT agility, 1–2
Alali, Ethar, 55
Algorithms, profiling developer skills, 144–145
Alignment
articulating strategy for general alignment, 99
IT business partner role in, 107–108
of IT with business, 101–103, 231
making business play its part, 107
mapping. See Alignment maps
MIT’s operating models and, 103–104
operational excellence, product leadership, and customer
timacy, 100–101
organizing for responsiveness over cost-efficiency, 229
pace-layered application strategy, 104
structural alignment, 105, 107
summary, 108
Alignment maps
  as information radiator, 234
  overview of, 104–106
  using for metrics, 174
Analytics, digital marketing and, 9.
  See also Metrics
Anonymity, confidential surveys and, 193
Anti-patterns
  appropriates uses of knowledge management team, 53
  spurious implementations of Agile development, 22
As-is (existing), customer journey maps, 8
Assessment
  Finland education example, 171–172
  metrics and, 173–175
  migrating from metrics-orientation to assessment-orientation, 179–180
  non-IT examples of principles in book, 243
Asynchronous communication, xxvii
Authority
  leading by influence not authority, 230
  unnecessary display of, 200
Automated rollbacks, improving effectiveness of delivery, 5
Autonomy
  absolute hierarchies and, 83
  authoritarian vs. professor entrepreneur model and, 83–85
  balancing with accountability, 75, 77
  centralization and, 42
  decentralization and, 46
  intrinsic motivation and, 33
  in IT service firms, 239
  norms and, 184
  outcome orientation allowing, 39–40
  removing targets to improve, 173
  self-organizing teams and, 168
  silos as symptom of wrong use of decentralization, 42–43
  team size and, 61
  what reduces/what enhances, 230
Availability, articulating strategies for, 99
Balanced scorecards, 162–163
Baselines
  for performance management, 163
  in validation of business cases, 116–117
Batch size, impacting cycle time, 50–51
BAU (business-as-usual) team, limitations of project teams, 139–140
Best fit over best tool as cultural norm, 185
  in tool evaluation, 157
Bezos, Jeff, 214
“Bimodal IT: How to Be Digitally Agile Without Making a Mess” (Gartner), 10
Binary code, 26–27
Blogs
  in internal communication, 206
  reinforcing norms, 185
Bottlenecks, theory of constraints, 125–126
Bring Your Own Device (BYOD), IT-I policies, 4
Budgeteering, 131
Budgeting
  Agile budgeting, 132
  budgeteering, 131
  business cases in, 115–116
  for capacity, not for projects, 110–112
Budgeting (continued)
collaborative approach, 133–134
cost/profit targets, 130–131
traditional approach, 130

Bugs
activity classification, 129–130
fixing in dev-box testing, 14–15
maintenance teams and, 65–66

Build vs. buy
articulating strategies, 99
buying software from ISVs rather than building, 138
outsourcing vs. in-house approaches, 240

Build-measure-learn feedback loop,
in Lean Startup, 117

Burnup charts, in measuring progress, 161

Business cases
deeemphasisizing financial business cases, 117
over-reliance on, 228
validation of, 115–116

Business-as-usual (BAU) team,
limitations of project teams, 139–140

Businesses
alignment with IT, 3–4,
101–103, 231
business attitude in GICs, 241
creating individual successes before higher-order success, 46
developing business capability into projects, 112–115

IT and, 3
IT effectiveness and, 5–6
making business play its part in success of IT efforts, 107

organizational agility touching all aspects of business, 7

Business-IT divide
elementary of functional silo, 42
overview of, 43
unbundling monolithic platform, 102–103
Buy vs. build. See Build vs. buy

BYOD (Bring Your Own Device),
IT-I policies, 4

Cain, Susan, 222

CAMS (culture, automation,
measurement, and sharing),
DevOps and, 16

Capabilities. See also Talent/skills
aligning with business, 112–113
assigning capability owners, 112
benefit of capability teams, 228
capability teams, 112–114
challenge of capability loss in long-lived teams, 141
defined, xxvii
in IT service firms, 240
outsourcing and, 114
platform of, 113
sudden projects and, 114–115
time required for developing, 139–140

Capability Maturity Model (CMM), 242

Capability roadmaps, 112–113

Capability teams
benefit of, 228
overview of, 112–114

Capacity
budgeting for capacity, not for projects, 110–112
setting work in progress limits, 122

CapEx (capital expenditure)
accounting without timesheets, 128–129
activity classification, 129–130
defined, xxvii
relationship to OpEx, 127–128
tracking, 228
Carrot-and-stick approach, vs.
intrinsic motivation, 179
CD. See Continuous delivery (CD)
CDO (Chief data officer), business-
IT divide and, 43
Center for Information Systems
Research, MIT Sloan School of
Management, 103
Centralization
conventional budgeting
and, 130
vs. decentralization, 41–42
impact on autonomy, 230
Change management
change programs and
initiatives, 121
order of adoption of Agile
principles, 233–234
Change responsiveness. See also
Responsiveness
over following a plan (Agile
principle), 14, 28
as solution to budgeteering, 131
Channel enhancement, as tactical
change, 8
The CHAOS Manifesto, 26, 121
Chargebacks
encouraging cost-efficiency over
responsiveness, 135
pros/cons in apportioning costs,
126–127
Checklists, preferred over document
or report templates, 232
Chief data officer (CDO), business-
IT divide and, 43
Chief information officer (CIO),
business-IT divide and, 43
Chief marketing officer (CMO),
business-IT divide and, 43
CIO (Chief information officer),
business-IT divide and, 43
Clients, involvement in IT services
ITS), 238–239
The Clock of the Long Now
(Brand), 104
CMM (Capability Maturity
Model), 242
CMO (chief marketing officer),
business-IT divide and, 43
Coaching leadership style, dealing
with opposition, 96
Cobudget tool, for collaborative
budgeting, 134
Code/coding, as development, 27
Collaboration
access control and, 149–150
budgeting and, 133–134
designing for unscripted
collaboration, 230–232
importance of individual
drive in, 33
intrinsic motivation and, 197
intrinsic motivation leading to
unscripted collaboration, 25,
34–35
matrix structure and, 70
reducing friction in shared service
interfaces, 55–56
in Schneider’s culture
model, 17
shift from control culture to
collaboration culture, 243
between teams vs. within teams,
48, 72
Collective ownership
difficulties with, 80–81
of outcomes across multiple
teams, 98
Commercial-off-the-shelf (COTS)
systems, in IT-I, 2
Communication
asynchronous, xxvii
blogs and videos in, 206
documents, reports, and
templates, 216–217
e-mail impacting balance with
work, 155–157
group meetings in, 205–206
Communication (continued)
interpersonal communication
problem mitigation, 203
interpersonal communication
problems, 198
intrinsic motivation and, 197–198
language’s impact on thinking, 154
meaning trumping aesthetics, 213–214
in new-hire orientation, 203
nonverbal microaggression, 199–200
online forums in, 206–208
overview of, 197
pitch culture and, 215–216
primacy of words/text in, 213
profiling developer skills, 144–145
pulling rank as problem in, 198–199
pulse charts in, 203–204
scaling employee engagement via
internal, 204–205
slide desks in, 214–215
summary, 217
surveys in, 206
verbal microaggression, 200–201
visual aids in, 211–213
war metaphor in business
language, 201–203
what reduces/what enhances
purpose, 231
written deliberation, 208–211
Communities of practice (CoP)
complementing outcome-oriented
organization, 73
enhancing mastery, 232
nurturing competencies, 65
Competency. See also Mastery
communities of practice
nurturing, 65
pros/cons of functional teams, 52–53
in Schneider’s culture model, 17
Competition
Agile culture and, 16–17
cooperation over, 186–187
Cone of uncertainty (Boehm), 211–213
Confidential surveys
benefits of, 193
ditching the pitch, 216
Consistency
over uniformity, 189–191
pros/cons of functional teams, 52
Continuous delivery (CD)
defined, xxvii
digital transformation and, 10
Gestalt Inc. example, 6
improving effectiveness of
delivery, 5
integral to Agile software delivery,
15–16
maintenance team limitations, 66
matrix structure and, 69
reducing cycle time, 72
Continuous Delivery (Humble and
Farley), 5
Continuous integration
decision record and, 88
defined, xxvii–xxviii
development and deployment
and, 5
fail-fast principle and, 18
integral to Agile software
development, 14–15
Contracts, IT services (ITS), 238
Control culture
microaggression and, 200
in Schneider’s culture model, 17
shift to collaboration culture, 243
Control mechanism, targets as,
166–167
Controllers, conventional budgeting
and, 130
Conway’s law, 72, 155
Cooperation
over competition, 186–187
rewards supporting, 187
CoP (communities of practice).
See Communities of
practice (CoP)
Cost centers
  GICs viewed as, 241
  vs. profit centers, 126
Cost targets, in budgeting, 130–131
Cost-benefit analysis, business cases in, 115
Cost-efficiency
  chargebacks encouraging over responsiveness, 135
  example of impact of communication protocol on responsiveness, 55–56
  IT services organizing for, 237
  operational excellence valuing, 100
  organizing for responsiveness over, 25, 30–32, 229
Costs, apportioning using
  chargebacks, 126
COTS (commercial-off-the-shelf) systems, in IT-I, 2
Credibility, planning and execution and, 94, 96
Cross-functional teams
  CapEx and OpEx and, 127–128
  defined, xxviii
  DevOps as, 58
  hospital pod example, 61–62
  matrix structures and, 71–72
  migrating to, 63–64
  museum layout example, 62
  non-IT examples of principles in book, 243
  open-plan office layout and, 220
  organizing for responsiveness, 58–59, 229
  overview of, 56–58
  preference for tools that blur boundaries between specialists, 153
  separation of duties in, 64
  size recommendations, 60–61
  specialization and, 60, 72
  staffing by skills, not by roles, 141–143
  taskonomy vs. taxonomy, 62–63
  utilization tradeoff, 60
Crowdsourcing, 185
Cultivation, in Schneider’s culture model, 17
Culture
  of accountability, 78
  Agile culture, 16–17
  cultural differences in GICs, 241
  DevOps stressing importance of, 16
  difficulty of changing, 243
  migrating from competition culture to cooperation culture, 187
  norms. See Norms
  organizing for responsiveness over cost-efficiency, 229
  permission-based cultures as risk-averse, 192
Culture, automation, measurement, and sharing (CAMS), DevOps and, 16
Customer collaboration, over contract negotiation (Agile principle), 14–15
Customer intimacy, 100–101
Customer journey maps, in business model innovation, 8
Customer relationships, 100–101
Cycle time
  batch size impacting, 50–51
  defined, xxviii
  issues in long development, 5–6
  pros/cons of functional teams, 52
  reducing, 72
  release interval compared with, 70
  value stream optimization and, 20
Dashboards, 162–163
Data-informed assessment, vs. data-driven measurement, 174
Decentralization
  vs. centralization, 41–42
Decentralization (continued)
designing for intrinsic motivation, 230
norms and, 184
positive actions in, 46
silos as symptom of wrong use of, 42–43

Decision making
accountability and, 78, 229
avoiding deadlocks, 94–95
decision record and, 88–89
overcoming deadlocks, 80
overview of, 86–87
pitch culture and, 215
resistance to decision record system, 91–92
scope of decision record system, 91
tools for decision record system, 89
transparency of decision records, 98
written deliberation and, 210

Decision records
accountability and, 88–89
continuous integration and, 88
organizing for responsiveness over cost-efficiency, 229
resistance to decision record system, 91–92
scope of decision record system, 90–91
tools for decision record system, 89–90

Delegation
accountability mapping and, 80
management styles and, 77
norms and autonomy and, 230

Delivery
continuous. See Continuous delivery (CD)
improving effectiveness of, 5
offshore centers. See GICs
profiling developer skills, 144–145
velocity metric applied to, 161

Denning, Steve, 13, 243, 244
Departmental (lower-level) silos, 44
Deployment pipeline, improving effectiveness of delivery, 5
Design
of better metrics, 175–178
framing the problem during team design, 47–48
importance of intrinsic motivation in, 25
for intrinsic motivation, 230
organizational. See Organizational design
of outcomes, 41
software development as design process, 28
source code as detailed form of, 26–27
taskonomy vs. taxonomy in human-centered design, 62–63
team design impacting responsiveness, 31–32
of teams, 47, 228
Desks, ergonomics of, 223
Dev-box testing, 14–15

Developers
benefits of DevOps to, 16
issues with Agile development and, 21–22
profiling skills of, 144–145

Development
contrasted with production, 27
cross-functional merger of development and IT operations, 58
IT-B contrasted with development organizations, 3
merging maintenance teams with development teams, 229
DevOps (development + operation)
CapEx and OpEx and, 127–128
cross-functional merger of development and IT operations, 58
defined, xxviii
digital transformation and, 10
maintenance team limitations, 66
reorganization required by, 1
supporting continuous delivery
via cross-pollination between
distinct IT operations, 16
Dibbits, Taco, 62
Differentiation
systems of differentiation or
engagement, xxix
utility vs. feature scope, 139
Digital business, xxviii
Digital marketing
digital transformation and,
7–9
specialty tools for, 153
Digital transformation
business-IT divide and, 43
cautions regarding large
projects, 122
defined, xxviii
order of adoption of Agile
principles, 233–234
tactical and strategic maneuvers
in, 7–10
Directing style, of leadership, 96
The Discipline of Market Leaders
(Treacy and Wiersema), 100
Documentation
of living policies, 188
role of e-mail in frivolous, 157
Documents, vehicles of
communication, 216–217
Dodds, Keith, 172
Drive (Pink), 33, 179, 194
E-mail
impacting balance of
communication and work,
155–157
in written deliberation, 210
Employees
engagement of, 204–205
new-hire orientation, 203
Empowerment, autonomy and, 33
End-of-life support, maintenance
teams and, 66
Engagement, of employees
scalability of, 204–205
systems of differentiation or
engagement, xxix
Enspiral, 133–134
Enterprise Architecture as
Strategy (Ross, Weill, and
Robertson), 103
Enterprise IT
business organization and, 3–4
IT-B and, 3
Entrepreneurs, as outcome owners,
84. See also Professor
and entrepreneur model
(Poppendieck)
Ergonomics, 222–223
Errors of commission, 86–87
Errors of omission, 86–87
Estimable, INVEST attributes, 22, 38
Excellence
aligning operational excellence,
product leadership, and
customer intimacy, 100–101
mastery as pursuit of, 34
operational excellence, 236–237
Execution. See also Planning and
evaluation
activities in, 92–93
disadvantages of separation of
planning from, 93–95
planners and, 93
Experience design (XD), xxix
Extreme programming (XP), 194
Extrinsic motivation
factors in, 33
targets and incentives in, 180
Face-to-face interactions, pros/cons
of remote office, 224
Fail-fast principle, in Agile
development, 18
Features, setting work in progress
limits, 122
Feedback
- fail-fast approach providing quick feedback, 18
- fast feedback from rolling approach to budgeting, 133
- in incremental development, 116–117
- in iterative development, 19
- metrics tracking, 162
- in mitigating microaggression, 203
- reducing friction in shared service interfaces, 55–56

Finance
- accounting without timesheets, 128–129
- activity classification, 129–130
- Agile approach to budgeting, 132
- budgeteering, 131
- CapEx and OpEx and, 127–128
- chargebacks, 126–127
- collaborative approach to budgeting, 133–134
- conventional budgeting, 130
- cost centers vs. profit centers, 126
- cost/profit targets in budgeting, 130–131
- overview of, 125
- relevance of, 125–126
- rolling approach to budgeting, 132–133
- summary, 134–135
- venture funding, 134

Financial business cases. See also Business cases
- assumptions and adjustments, 115
deephasizing, 117
Forecasts, metrics and, 160
Forgiveness, vs. permission, 192
Forums, online forums for internal communication, 206–208
Freeman, Jo, 76
Full-time equivalents (FTEs), in matrix with dedicated capacity and fungible people, 70

Function leads
- with autonomy but not authority, 98
- comparing authority models, 83–85
defined, xxviii
- leading by influence not authority, 230
- in matrix organizations, 82–83
- minimizing power struggles by designation of outcome owners, 82
- pairing with outcome owners, 86
- preventing territorial conflicts in absolute hierarchies, 83

Functional organizations
- activity-oriented teams and, 48–49, 52–53
- matrix structures and. See Matrix structures
- what reduces/what enhances mastery, 232

Functional silos. See Silos

Games/gamification
- disadvantages of competition, 187
targets and, 168–170

GICs
- business attitude in, 241
cultural differences and, 241
future trends and prognosis, 242–243
- legacy of Capability Maturity Model impacting, 242
management styles and, 241
overview of, 240

Global optimum, vs. local optima, 164–166

Goodhart, Charles, 170
Goodhart’s Law (of economic policy targets), 170
Google Moderator, online forums for internal communication, 208
Governance, functions of governance teams, 120–121
Granof, Phil, 203
Graphics, use/misuse of visual aids, 211
Group meetings, in internal communication, 205–206
Guidelines, preferred over document or report templates, 232

Handoffs
defined, xxviii
functional organizations and, 59
high-latency handoffs impacting activity-oriented teams, 49–51
maintenance teams and, 73
matrix structure and, 69
organizing for responsiveness over cost-efficiency, 229
specialization and, 72
Hardening phase, not required in continuous delivery, 15
Hearsay, avoiding culture of, 193
Hierarchies
absolute, 83
communication protocols and, 199
proper use of, 76–77
what reduces/what enhances purpose, 231
Highest-paid person’s opinion (HiPPO), 88
Hospital pod, example of cross-functional organization, 61–62
How Buildings Learn (Brand), 104
Hsieh, Tony, 32
Human-centered design, 62–63
Humble Inquiry (Schein), 203
IdeaBoardz, online forums for internal communication, 208
In market time, questioning functional organizations and, 52
Inbox, how e-mail technology shapes us, 156
Incentives
getting rid of, 172–173
limitations of, 163–164
moving from if-then approach to now-that approach, 179
targets implying, 171
undermining intrinsic motivators, 167–168
Incremental development
fast feedback in, 116–117
iterative development preferred over, 18–20
Independent, negotiable, valuable, estimable, small, and testable (INVEST)
negotiable attribute (N), 22
qualities of good stories, 38
Independent software vendors. See ISVs (independent software vendors)
Independent testing, appropriates uses of activity-oriented teams, 53–54
Influence
leading by influence not authority, 230
power and, 75
professor entrepreneur model and, 84
Informal power structures, problems with, 98
Information radiators
accountability maps as, 98
in Agile development, 20–21
alignment maps as, 104–106
chart of, 234
office wall space requirements for, 220
proper use of targets and, 171
pulse charts as, 203–204
story boards in setting work in progress limits, 123
Infrastructure on demand, improving effectiveness of delivery, 5
Interactive voice response (IVR), 30
Interdisciplinary team. See Cross-functional teams
Internal communication
  blogs and videos in, 206
  group meetings in, 205–206
  online forums in, 206–208
  scaling employee engagement via, 204–205
  surveys in, 206
Internal scope
  budgeting for capacity, not for projects, 111
  defined, xxviii
  in iterative development, 19
  variable nature in value-driven development, 30
Internet businesses
  blurring of lines between online and offline transactions, 9
  defined, xxviii
  IT and, 3–4
  IT-B and, 3
Interpersonal communication
  new-hire orientation, 203
  nonverbal microaggression, 199–200
  problem mitigation, 203
  problems, 198
  pulling rank as problem in, 198–199
  pulse charts, 203–204
  verbal microaggression, 200–201
  war metaphor in business language, 201–203
  what reduces/what enhances purpose, 231
Intrinsic motivation
  autonomy and empowerment and, 33
  vs. carrot-and-stick approach, 179
  centralization impacting, 42
  communications and, 197–198
  designing for, 230–232
  IT services (ITS) and, 239
  leading to unscripted collaboration, 25, 34–35
  mastery and purpose and, 34
  organic approach to creating culture of, 35
  outcome ownership giving team sense of purpose, 39
  removing targets to improve, 173
  targets and incentives undermining, 167–168
  targets leading to gaming, 168–170
  “Inverse Conway Maneuver,” 72
  INVEST (independent, negotiable, valuable, estimable, small, and testable)
    negotiable attribute (N), 22
    qualities of good stories, 38
  ISVs (independent software vendors)
    buying vs. building and, 138
    defined, xxviii
    IT and, 3–4
  IT business partner role, 107–108
  IT Business Partnerships (Topinka), 102
IT organization
  aligning IT with business strategy, 101–103
  business organization and, 3–4
  business-IT divide and, 43
  DevOps and, 58
  effectiveness of, 5–6
  merging business and IT, 3–4
  mixing IT-I and IT-B in, 2–3
  non-IT examples of principles in book, 243
  translating team-level agility into IT agility, 1–2
  two-speed approach to (bimodal), 10
IT services (ITS)
  applying book principles to, 236–237
  client involvement, 238–239
  contracts, 238
  end-user access, 238
  future trends, 240
  intrinsic motivation and, 239
IT-B (build and operate)
budgeting for capacity, not for projects, 110–112
cost centers vs. profit centers, 126
defined, xxix
merging business and IT, 4
migrating from IT-B matrix to
cross-functional team, 63–64
migrating from IT-B matrix to
outcome-oriented team, 73
mixing IT-I and IT-B in IT
organization, 2–3
outsourcing, 66–68
product leadership driving, 100
responsiveness vs. cost-efficiency
and, 30–31
structural alignment, 105, 107
unbundling monolithic platform,
102–103
Iterative development
governing for value over
predictability, 28
preferred over incremental
development, 18–20
value stream optimization and, 20
IT-I (infrastructure and pure
operations)
defined, xxix
merger of business and IT and, 4
mixing IT-I and IT-B in IT
organization, 2–3
IVR (Interactive voice response), 30
Job titles, limiting nature of, 143
Just-in-time features, in iterative
development, 19
Kanban boards
as information radiator, 234
setting work in progress limits,
122–123
Key performance indicators
(KPIs), 112
Key responsibility areas (KRAs), 78
Knowledge management (KM)
team, 53
Kohn, Alfie, 187
Koparati Inc., 101
Kranzberg, Melvin, 154
Labor, labor-intensive nature of IT, 4
Lagging indicators, designing better
metrics, 177
Language, impact on thinking, 154
Laptop computers, ergonomics of,
222–223
Layout, office. See Open-plan layout
The Leader’s Guide to Radical
Management (Denning), 244
Leadership
aligning operational excellence,
product leadership, and
customer intimacy, 100–101
clarifying ownership, 81
hierarchy and, 76
IT services and, 237
outcome ownership and, 40
role of executive leadership in
decision record system, 91
styles in dealing with
opposition, 96–97
Leads. See Function leads
Lean
build-measure-learn feedback loop
in Lean Startup, 117
product discovery techniques, 10
waiting lanes (queues) and active
lanes (work centers), 129
The Lean Startup (Ries), 176
Leveraged teams, in IT services, 237
Line functions, organizing by
function, 48–49
Lines of business (LOBs), creating
individual successes before
higher-order success, 46
Lobbying culture, 215
LOBs (lines of business), creating
individual successes before
higher-order success, 46
Local optima
  vs. global optimum, 164
IT services example, 238
  in space and in time, 165–166
Long-lived teams
  challenges of/objections to, 141
costs of, 140
  enhancing mastery, 232
IT services and, 240
Loomio tool, for decision record
  system, 89
Mailing lists, for internal
  communication, 208
Maintenance teams
  handoffs and, 73
  merging with development
teams, 229
  overview of, 65–66
Make vs. buy. See Build vs. buy
Management
  assessment and, 173–174
effects of management styles, 77
  in GICs, 241
Manager-as-supervisor model, 34
Managers
  in activity-oriented teams, 49
  monolithic cross-functional teams
  and, 71
  project managers, 119–120
Maps/mapping
  accountability. See Accountability
  maps
  alignment. See Alignment maps
  balancing roadmap fulfillment
  with responsiveness, 31
  capability roadmaps, 112–113
  customer journey maps, 8
Marketing, digital transformation
  and, 7–9
Mastery
  communities of practice
  nurturing, 65
  intrinsic motivation and, 34
  in IT service firms, 239
  what reduces/what
  enhances, 232
Matrix paralysis, 82–83
Matrix structures
  accountability and, 82–83
cross-functional teams and, 71–72
  with dedicated capacity and
  fungible people, 70
  with dedicated capacity and
  people, 70
  functional organizations and, 72
  migrating to outcome-oriented
  team, 63–64, 73
  migrating to professor-
  entrepreneur model, 86
  pros/cons of, 68–69
  shared services as, 69–70
McLuhan, Marshall, 154, 156, 215
Meaning, trumping aesthetics in
  visual aids, 213–214
Measurable, qualities of good
  tasks, 38
Meetings
  group, 205–206
  one-on-one hearings (meetings), 193
Metrics
  assessment and, 173–175, 243
  compensating metrics, 177–178
  dashboard limitations, 162–163
  dealing with unknown unknowns,
  161–162
  designing, 175–177
  getting rid of incentives, 172–173
  Goodhart’s law of economic
  policy targets, 170
  implicit targets, 170–171
  as information radiator, 234
  local optima vs. global optimum,
  164–166
  measurement vs. targets and
  incentives, 163
  measuring but not
  forecasting, 160
  migrating to assessment-
  orientation, 179–180
not the whole story, 159–160
objections to reforming, 178–179
outcome-oriented metrics, 112
overview of, 159
predictability vs. adaptability, 228
reforms to, 171–172
removing targets, 173
summary, 180–181
target and incentive limitations, 163–164
targets and incentives undermining intrinsic motivators, 167–168
targets as control mechanism, 166–167
targets implying incentives, 171
targets leading to gaming, 168–170
velocity use as, 161
Microaggression
impact on purpose, 231
mitigating, 203–204
nonverbal, 199–200
verbal, 200–201
Micromanagement, role of targets in, 167
Migration
to assessment-orientation, 179–180
to cooperation culture, 187
to cross-functional teams, 63–64
order of adoption in Agile, 233–234
to professor-entrepreneur model, 86
Minimum viable argument test, in written deliberation, 209
Mini-waterfall development, 19
Mistakes
disaster recovery example, 87–88
errors of commission and omission, 86–87
learning from, 87
MIT Sloan School of Management, 103
Mobile analytics, digital marketing and, 9
Mobile-first strategies, 8
Monolithic cross functional teams, 71–72
Motivation
absolute hierarchies and, 83
extrinsic, 33
intrinsic. See Intrinsic motivation
targets and incentives impacting, 164
Multifunctional teams. See Cross-functional teams
Museum layout, example of cross-functional organization, 62
Mutually exclusive goal attainment programs (Kohn), 187
Need-to-know, access control options, 149
Need-to-restrict, access control options, 149–150, 230
Need-to-use, access control options, 149, 230
Negotiable
attributes of good stories, 22
qualities of good outcomes, 38
testing sub-outcomes, 46
New-hire orientation
enhances purpose, 231
mitigating interpersonal communication problems, 203
Nonverbal microaggression, 199–200
Norman, Dan, 62
Norman, Donald, 190
Norms
asking for forgiveness, not for permission, 192
balancing theory and practice, 193–195
confidential surveys and, 193
consistency over uniformity, 189–191
cooperation over competition, 186–187
creating living policies, 187–188
impact on autonomy, 230
mechanics of reinforcing, 185
overview of, 183–184
summary, 195
what they are, 184–185

Office
benefits of open-plan layout, 219–220
criticism of open-plan layout, 222
ergonomics and, 222–223
overview of, 219
solitude and privacy and, 221–222
summary, 225–226
wall space in open layouts, 220–221
working remotely, 224–225

Offshore delivery centers. See GICs

One-on-one hearings (meetings), 193

Online forums
enhances purpose, 231
in internal communication, 206–208

Open-plan layout
accommodating solitude and privacy, 221–222
benefits of, 219–220
criticism of, 222
open seating, 220
wall space in, 220–221
what reduces/what enhances mastery, 232

Operating expenditure. See OpEx
(operating expenditure)

Operational excellence
aligning with product leadership and customer intimacy, 100–101
IT services focus on, 236–237

OpEx (operating expenditure)
accounting without timesheets, 128–129
activity classification, 129–130
defined, xxix
relationship to CapEx, 127–128
tracking, 228

Optimum outcome, measuring local optima vs. global optimum, 164–166

Org chart debt, 97–98

Organizational design
expansive view of, 2, 227
topics in, 10–11

Organizational information radiators. See Information radiators

Organizational policies, living policies, 187–188

Outcome owners
accountability and, 78–79
assigning accountability, 81
choosing, 85–86
on cross-functional teams, 57
defined, xxix
entrepreneurs as, 84
leadership and responsiveness and, 40
minimizing power struggles by designation of, 82
pairing with function leads, 86
Swedish bank example, 132
targets undermining ownership, 168

Outcome-oriented teams
compared with activity-oriented teams, 43
defined, xxix
outcome ownership and, 40, 46
purpose enhanced by, 231
shifting from activity-oriented teams to, 73
specialization in, 39–40

Outcomes
accountability mapping and, 79–80
business activities and, 37–38
Index

centralization vs. decentralization, 42
defined, xxix
designing, 41
determining ownership, 78–79
dividing into suboutcomes, 73
factors in successful, 6
forming teams around outcomes not activities, 46
independent value of, 45–46
outcome orientation allowing autonomy, 39–40
outcome-oriented metrics, 112, 176
outsourcing along, 67–68
ownership of, 40, 57
productivity vs. outcome realization, 228
qualities of good outcomes, 38
reasons why multiple teams are responsible for single outcome, 47–48
reducing need for targets, 230
team size and, 61
value stream optimization, 20
Outsourcing
build vs. buy and, 240
capability teams and, 114
dealing with talent crunch, 137–138
Dell business case analysis of, 116
GICs compared with, 240
IT services and, 238
IT-B work, 66–68

Pairing part-time staff with full-time staff, 232
Payment Card Industry Data Security Standard (PCI-DSS), 64
Peer reviews, 193
Performance
assessing. See Assessment
measuring. See Metrics
micromanagement and, 167
target and incentive limitations for measuring, 163–164
Permissive culture
asking for forgiveness, not for permission, 192
organizing for responsiveness over cost-efficiency, 229
Personality mix, on teams, 146
Pilot team, 64
Pitch culture, 215–216
Plan-driven development
difficulty of tracking benefits in, 116
limitations of, 109–110
project managers in, 119–120
value-driven approach compared with, 29–30, 228
Planning and execution
combining enhances mastery, 232
dealing with opposition in planning process, 96–97
disadvantages of separating, 93–95
overlapping, 96
overview of, 92–93
planners role in execution, 93
separating planning from execution, 120
PMO (project management office), in IT-B, 3
Policies
creating living, 187–188
parochialism (Rieger), 191
purpose and, 231
secondary to business goals, 189

Part-time work
avoid part-time assignments, 145–146

Pace-layered application strategy, 104
Paradigm shift (Kuhn), 243
Parochialism (Rieger), 191
Participation. See Engagement, of employees
Partitions, in open-plan office, 220–221

Policies
creating living, 187–188
parochialism (Rieger), 191
purpose and, 231
secondary to business goals, 189

Outsourcing
build vs. buy and, 240
capability teams and, 114
dealing with talent crunch, 137–138
Dell business case analysis of, 116
GICs compared with, 240
IT services and, 238
IT-B work, 66–68

Payment Card Industry Data Security Standard (PCI-DSS), 64
Peer reviews, 193
Performance
assessing. See Assessment
measuring. See Metrics
micromanagement and, 167
target and incentive limitations for measuring, 163–164
Permissive culture
asking for forgiveness, not for permission, 192
organizing for responsiveness over cost-efficiency, 229
Personality mix, on teams, 146
Pilot team, 64
Pitch culture, 215–216
Plan-driven development
difficulty of tracking benefits in, 116
limitations of, 109–110
project managers in, 119–120
value-driven approach compared with, 29–30, 228
Planning and execution
combining enhances mastery, 232
dealing with opposition in planning process, 96–97
disadvantages of separating, 93–95
overlapping, 96
overview of, 92–93
planners role in execution, 93
separating planning from execution, 120
PMO (project management office), in IT-B, 3
Parochialism (Rieger), 191
Purpose and, 231
Secondary to business goals, 189
Poly-skilled teams. See Cross-functional teams
Portfolio wall, as information radiator, 234
Posture, ergonomics and, 222–223
Power
  authoritarian vs. professor entrepreneur model and, 83–85
  hierarchy and, 75–77
  minimizing power struggles, 82
  power struggles in matrix organizations, 82–83
  single power center, 85
  unnecessary power symbolism, 200
Power vacuum, 76
Practice, balancing theory with, 193–195
Predictability
  adaptability metrics preferred over predictability metrics, 176–177
  chasing value over predictability, 25, 28–30, 228
  dealing with unknown unknowns, 161–162
  project and release plans and, 109–110
Principles, living policies and, 187
Privacy, office layout and, 221–222
Processes
  living policies and, 187
  secondary to business goals, 189
Product leadership, 100–101
Product silos, 43. See also Silos
Production, contrasted with development, 27
Productivity, vs. outcome realization, 228
Products
  activity-oriented teams and, 49
  aligning operational excellence, product leadership, and customer intimacy, 100–101
  creating individual successes before higher-order success, 46
  migrating from IT-B matrix to cross-functional team, 63–64
  pros/cons of functional teams, 52
  source code vs. binary code, 26–27
  valuing over projects, 120
  what user or client uses, 27–28
Professor and entrepreneur model (Poppendieck)
  avoiding territorial struggles, 98
  comparing authority models, 84–85
  decision record and, 88
  migrating to, 86
  overview of, 59
Profit centers, vs. cost centers, 126
Profit targets, in budgeting, 130–131
Programming languages, profiling developer skills, 144–145
Project management office (PMO), in IT-B, 3
Project managers, 119–120
Project plans, predictive nature of software development and, 26
Project teams
  costs of long-lived teams vs., 140
  limitations of, 139–140
Projects
  budgeting for capacity, not for projects, 110–112
  business cases, 115–116
  change programs and initiatives, 121
  de-emphasizing financial business cases, 117
  developing business capability, 112–115
  digital transformation programs, 122
  governance, 120–121
  limitations of plan-driven, 109–110
  overview of, 109
  project managers, 119–120
  project-based budgeting, 228
setting work in progress limits, 122–123
summary, 123–124
validating business case benefits, 116–117
value-driven, 117–119
Pulling rank, interpersonal communication problems, 198–199
Pulse charts
   enhancing purpose, 231
   as information radiator, 234
   mitigating interpersonal communication problems, 203–204
Purpose
   intrinsic motivation and, 34
   in IT service firms, 239
   loss of purpose in shared service approach, 55
   outcome ownership giving team sense of, 39
   team size and, 61
   what reduces/what enhances, 231
Queues (waiting lanes), in Lean terminology, 129
Quiet (Cain), 222
RAG (red, amber, or green) reports, 174–175
Recruitment. See also Staffing
gaming and, 169
labor-intensive nature of IT and, 4
skill profiles in, 145
Red, amber, or green (RAG) reports, 174–175
Regional silos, 43
Reinventing Organizations, 244
Releases
   continuous delivery and, 15–16
   predictability of release plans, 109–110
   release interval compared with cycle time, 70, 72
Relevant, qualities of good tasks, 38
Remote (home) office, pros/cons of, 224–225
Reorganization, examples of
   structural reorganization, 97
Reports
   red, amber, or green (RAG) reports, 174–175
   use/misuse of visual aids, 211
   vehicles of communication, 216–217
Respect, disadvantages of separation of planning from execution, 93–94
Responsiveness
   accountability and, 75
   balancing roadmap fulfillment with, 31
   chargebacks encouraging cost-efficiency over, 135
   in city hospital study, 61
   cross-functional teams organizing for, 58–59
   IT services organizing for cost-efficiency over, 237
   matrix structure and, 70
   organizing for responsiveness over cost-effectiveness, 25, 30–32, 72, 229
   outcome ownership and, 40
   team design impacting, 31–32
   team size and, 61
   tradeoff with over utilization of specialists, 229
Retention, of staff, 139
Rewards, targets and, 186. See also Incentives
Rijksmuseum, 62
Risk-aversion, permission-based cultures as, 192
Roadmaps. See Maps/mapping
Rogue IT. See Shadow IT
Roles
- clarity of, 81–82
- IT business partner role, 107–108
  for planning and execution. See
  Planning and execution
- staffing by skills, not by roles, 141–143, 232
Rolling approach, in Agile budgeting, 132–133
Rose, Charlie, 214
SaaS (software-as-a-service)
- defined, xxix
- merger of business and IT and, 4–5
Scalability
- of access control options, 150
- of conversations in context, 178–179
- devolution of control and, 168
- of employee engagement, 204–205
- problem with premature scaling of
  Agile development, 21
Scaled Agile Framework, 111, 243
Schneider’s culture model, 17, 173
Scope
- budgeting for capacity, not for
  projects, 110–112
- of decision record system, 91
- internal scope, xxviii
- limiting scope and sophistication
  of solutions, 138–139
Scrum
- activity-oriented teams in, 22
- balancing theory and practice, 194
Separation of duties
- in cross-functional teams, 64
- separating planning from
  execution, 120
SEPG (software engineering process
group), 242
Shadow IT
- hiring IT contractors and
  leveraging the cloud, 131
- merging business and IT, 4–5
Shared services
- limitations of, 73
- loss of purpose due to, 55
- as matrix structure, 69–70
- reducing friction in shared service
  interfaces, 55–56
- as type of activity-oriented team, 54–55
Showcases, fail-fast principle and, 18
Silos
- in business-IT divide, 43
- combining higher-order business
  outcomes, 44–45
- defined, xxix
- due to tool access, 151–152
- due to tool specialization, 153
- due to tool usage, 152–153
- functional, 42
- overview of, 42–43
Six laws of technology (Kranzberg), 154–155
Skills. See Capabilities; Talent/skills
Slide decks
- ditching the pitch, 215–216
- non-IT examples of Agile
  principles, 243
- use/misuse of visual aids, 214–215
Slide presentations, use/misuse of
visual aids, 211
SMART (specific, measurable,
achievable, relevant, and
time-boxed), qualities of good
tasks, 38
Social systems
- in organizations, 82
- software development as social
  activity, 159
Software development
Agile approach to. See Agile
Manifesto
- budgeting for capacity, not for
  projects, 110–112
- continuous integration in,
  14–15
- as design process, 28
limitations of plan-driven projects, 109–110
measuring but not forecasting, 160
product is what user or client uses, 27–28
reconsidering based on principle of value over predictability, 25–26
as social activity, 159
source code not binary code product of, 26–27
Software engineering process group (SEPG), 242
Software quality analysts (SQA), 242
Software-as-a-service (SaaS)
   defined, xxiv
   merger of business and IT and, 4–5
Solitude, office layout and, 221–222
Source code, as product of software development, 26–27
SOX regulations, separation of duties and, 64
Specialist leads. See Function leads
Specialist teams. See Activity-oriented teams
Specialists
   avoid part-time assignments, 145–146
   communities of practice nurturing competency, 65
cross-functional teams and, 60
organizing for responsiveness over cost-efficiency, 229
preference for tools that blur boundaries between, 153
Specialization
cross-functional teams and, 56, 59
handoffs resulting from, 49–51
in matrix organizations, 82
organizing teams by function, 49
pros/cons of functional teams, 52
silos due to tool specialization, 153
teams and, 72
underutilization of specialists on cross-functional teams, 60
Specific, measurable, achievable, relevant, and time-boxed (SMART), qualities of good tasks, 38
Spolsky, Joel, 222
Spotify, example of cross-functional organization, 59
Spurious implementations, in Agile development, 22
SQA (software quality analysts), 242
Stack Exchange-style, web-based Q&A forum, 208
Staffing
   avoid part-time assignments, 145–146
   challenges of and objections to long-lived teams, 141
costs of long-lived teams vs. project teams, 140
dealing with talent crunch, 137–138
   full-time equivalents (FTEs), 70
IT services (ITS) and, 237, 240
job titles, 143
labor-intensive nature of IT, 4
limitations of project teams, 139–140
limiting scope and sophistication of solutions, 138–139
organizing by function, 48–49
overview of, 137
personality mix on teams, 146
profiling skills, 144–145
retaining staff, 139
by skills, not by roles, 141–143, 232
summary, 146–147
Standardization
   impact on autonomy, 230
operating models guiding, 103–104
parochialism (Rieger) and, 191
pros/cons of functional teams, 52
tool use and, 158
Standing desks, ergonomics and, 223
Startup funding, 134
Stories
  outcomes and activities and, 37–38
  in propagating norms, 185
  qualities of good stories, 22, 38
Story boards
  accounting without using
timesheets, 128–129
  setting work in progress limits, 123
Strategic IT, IT-B contrasted with, 3
Strategies
  aligning IT with business strategy,
  101–103
  articulating general alignment
  strategy, 99
  changing to operational processes
  and IT systems, 8
  operating models in, 103–104
  pace-layered application
  strategy, 104
Structural alignment, 105, 107
Structural Reorganization, 97–98
Superstructure, of organizations
  activities and outcomes and,
  37–39
  centralization and
decentralization, 41–42
  outcome design, 41
  outcome orientation allowing
  autonomy, 39–40
  outcome ownership, 40
  overview of, 37
  silos, 42–43
  silos combining higher-order
  business outcomes, 44–45
  silos in business-IT divide, 43
  summary, 45–46
Surveys
  confidential, 193, 216
  in internal communication, 206
Tasks, qualities of good tasks, 38
Taxonomy, vs. taskonomy, 62–63
Teams
  activity-oriented teams, xxvii,
  48–49
  appropriates uses of activity-
  oriented teams, 53
  avoid part-time assignments,
  145–146
  capability teams, 112–113
  challenges of and objections to
  long-lived teams, 141
  communities of practice and, 65
costs of long-lived teams vs. project teams, 140
cross-functional teams, xxviii, 56–58
danger of becoming silos, 45
design impacting responsiveness, 31–32
designing, 47
DevOps as cross-functional team, 58
forming around outcomes not activities, 46
framing the problem when designing, 47–48
functional organizations and, 52–53
governance teams, 120–121
high-latency handoffs impacting activity-oriented teams, 49–51
hospital pod example of cross-functional team, 61–62
independent testing, verification, and validation, 53–54
IT services (ITS) and, 237
limitations of project teams, 139–140
loss of purpose in shared service approach, 55
maintenance teams, 65–66
matrix structure and, 68–72
merging maintenance with development, 229
migrating to cross-functional teams, 63–64
museum layout as example of cross-functional organization, 62
organizing for responsiveness in cross-functional teams, 58–59
outcome-oriented teams, xxix
outcome-oriented vs. activity-oriented teams, 43
outsourcing and, 66–68
personality mix on, 146
reducing friction in shared service interfaces, 55–56
self-organizing teams and autonomy, 168
separation of duties in cross-functional teams, 64
shared services compared with activity-oriented teams, 54–55
size recommendations, 60–61
specialists limitations on cross-functional teams, 60
summary, 72–73
taskonomy vs. taxonomy, 62–63
translating team-level agility into IT agility, 1–2
utilization tradeoff in cross-functional teams, 60
Technology
profiling developer skills, 144–145
role in creation of silos, 154–155
Templates, vehicles of communication, 216–217
Test automation, improving effectiveness of delivery, 5
Testable, INVEST attributes, 22, 38
Testable, valuable, independently achievable, and negotiable (TVIN)
qualities of good outcomes, 38
testing sub-outcomes, 46
Test-driven development, improving effectiveness of delivery, 5
Tests
Dev-box testing, 14–15
independent testing, verification, and validation, 53–54
Text, primacy in visual aids, 213
Theory, balancing with practice, 193–195
Theory of constraints, 125–126
Theory of group development (Tuckman), 139–140
ThoughtWorks, 172
Three-horizons framework (McKinsey), 85–86
Time-boxed, qualities of good tasks, 38
Timesheets, accounting without using, 128–129
To-be (enhanced) journey maps, 8
To-market time, in functional organizations, 52
Tooling
  accountability tools, 89
  controlling access on need-to-restrict basis, 149–150
  e-mail impacting balance of communication and work, 155–157
  evaluating, 157–158
  impact of standardization on autonomy, 230
  overview of, 149
  silos due to tool access, 151–152
  silos due to tool specialization, 153
  silos due to tool usage, 152–153
  summary, 158
  technology in creation of silos, 154–155
Training, dealing with talent crunch, 137–138
Transparency, confidential surveys and, 193
Trial-and-error methods, 194
Trust, freedom and, 183–184
T-shaped people
  cross-functional teams and, 60
  job titles and, 143
  staffing by skills, not by roles, 141
TVIN (testable, valuable, independently achievable, and negotiable)
  qualities of good outcomes, 38
  testing sub-outcomes, 46
Two-pizza team, team size and, 60
UAT (user acceptance tests), 53–54
Uniformity, consistency over, 189–191
Unit level policies, 188
Unscripted collaboration. See also Collaboration
  defined, xxix
  designing for, 230–232
  intrinsic motivation leading to, 25, 34–35
The Upside of Turbulence (Sull), 244
Usage analytics, in validation of business cases, 116–117
Use cases, in iterative development, 19
User acceptance tests (UAT), 53–54
User experience. See UX (user experience)
Users, end-user access in IT services, 238
Utilization
  IT services focus on, 237
  pros/cons of functional teams, 52
  tradeoff in cross-functional teams, 60
UX (user experience)
  appropriate uses of activity-oriented teams, 53
  consistency over uniformity, 191
  defined, xxix
  outcome design and, 41
Validation
  appropriate uses of activity-oriented teams, 54
  of benefits preferred to over-reliance on business cases, 228
  of business cases, 116–117
Valuable, INVEST attributes, 22, 38
Value
  chasing value over predictability, 28, 228
  qualities of good outcomes, 38
  reconsidering software development based on principle of value over predictability, 25
  testing sub-outcomes, 46
Value neutrality argument, 163
Value streams
  defined, xxix
high-latency handoffs impacting activity-oriented teams, 49–50
optimization of, 20
Value-driven development
benefits of, 228
comparing with plan-driven approach, 29–30, 117–119
project managers in, 119–120
Vanity metrics (Ries), 176
Velocity
designing better metrics, 176
lagging indicators, 177
measuring functionality delivered over time period, 161
works as measurement but not as target, 167–168
Venture funding, 134
Verbal microaggression, 200–201
Verification, appropriate uses of activity-oriented teams, 54
Version-controlled configuration, improving effectiveness of delivery, 5
Videos, in internal communication, 206
Violence, war metaphor in business language, 201–203
Visual aids
meaning trumping aesthetics, 213–214
overview of, 211–213
pitch culture and, 215–216
primacy of words/text in, 213
slide desks in, 214–215
what reduces/what enhances mastery, 232
Wages, IT-B costs, 111
Waiting lanes (queues), in Lean terminology, 129
Wall space, in open office layouts, 220–221
War metaphor, in business language, 201–203
Web analytics, digital marketing and, 9
Web-based discussion forums, 208
Wikipedia
example of consistency over uniformity, 191
example of inverse access control, 150
Wikis, example of inverse access control, 150
WIP (work in progress), limiting, 122–123
Words/text, primacy in visual aids, 213
Work, e-mail impacting balance of communication and work, 155–157
Work centers (active lanes), in Lean terminology, 129
Work ethic, cultural norms and, 183
Work in progress (WIP), limiting, 122–123
Written deliberation, 208–211
XD (experience design), xxix
XP (extreme programming), 194
Zappos, example of organizing for responsiveness over cost-efficiency, 32
Zero downtime deployment, improving effectiveness of delivery, 5
LearnIT at InformIT


- Learn about new releases and special promotions by subscribing to a wide variety of monthly newsletters. Visit informit.com/newsletters.
- FREE Podcasts from experts at informit.com/podcasts.
- Read the latest author articles and sample chapters at informit.com/articles.
- Get Advice and tips from expert blogs at informit.com/blogs.

Visit informit.com to find out all the ways you can access the hottest technology content.

Are you part of the IT crowd?

Connect with Pearson authors and editors via RSS feeds, Facebook, Twitter, YouTube and more! Visit informit.com/socialconnect.
Register the Addison-Wesley, Exam Cram, Prentice Hall, Que, and Sams products you own to unlock great benefits.

To begin the registration process, simply go to informit.com/register to sign in or create an account. You will then be prompted to enter the 10- or 13-digit ISBN that appears on the back cover of your product.

Registering your products can unlock the following benefits:

- Access to supplemental content, including bonus chapters, source code, or project files.
- A coupon to be used on your next purchase.

Registration benefits vary by product. Benefits will be listed on your Account page under Registered Products.

About InformIT — THE TRUSTED TECHNOLOGY LEARNING SOURCE

INFORMIT IS HOME TO THE LEADING TECHNOLOGY PUBLISHING IMPRINTS Addison-Wesley Professional, Cisco Press, Exam Cram, IBM Press, Prentice Hall Professional, Que, and Sams. Here you will gain access to quality and trusted content and resources from the authors, creators, innovators, and leaders of technology. Whether you’re looking for a book on a new technology, a helpful article, timely newsletters, or access to the Safari Books Online digital library, InformIT has a solution for you.