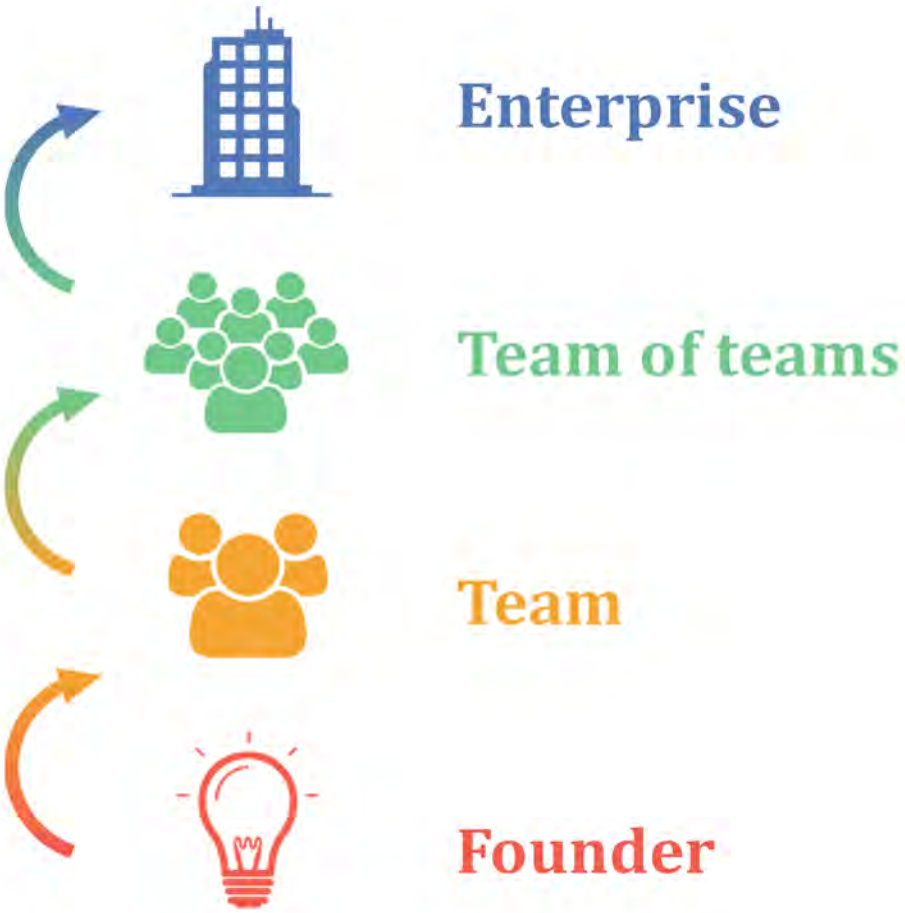
Digital delivery



Concepts and practices

Charles T. Betz

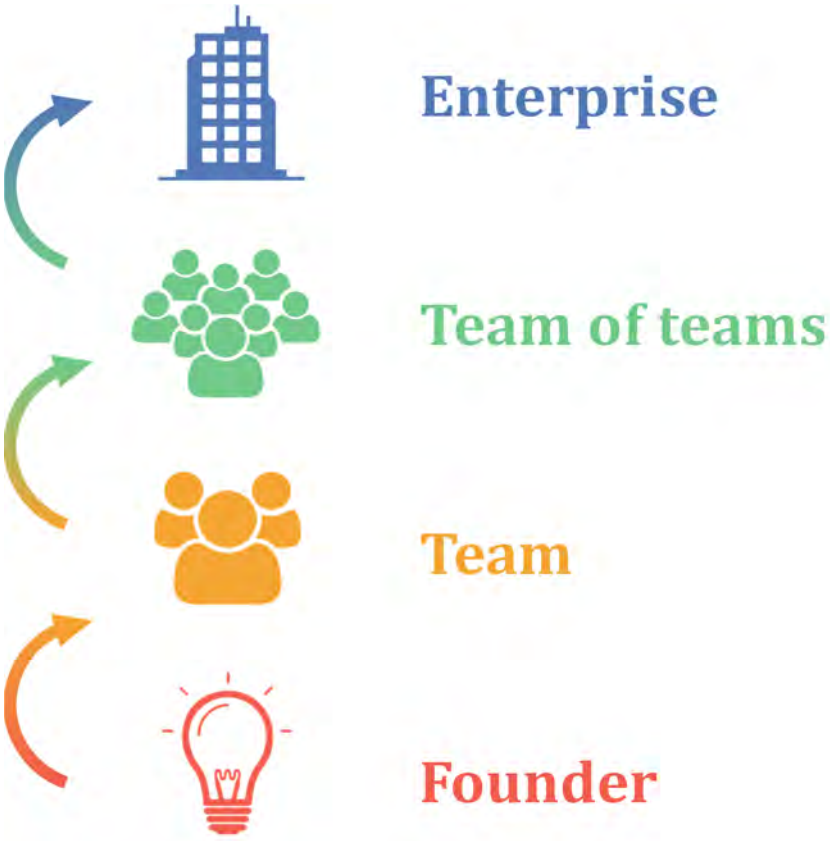
Praise for this book

Digital Delivery is a perfect fit for my Management Information Systems class to introduce students to the fast-paced world of IT Infrastructure that they will be dealing with shortly upon graduation. This book uses multiple perspectives (Founder, Team Leader, VP, C-level executive) to demonstrate to the student not only how a business grows, but how they need to continually grow their skill set. The use of hands-on exercises encouraged by the format of this book complements my teaching style that allows students to learn by doing, failing and doing again. An additional benefit is that this book begins with a focus on the startup mentality which I will use in my Business Innovation class.

Prof. Pat Paulson, Winona State University

Digital Delivery

Concepts and practices


Enterprise
Team of teams
Team
Founder

Digital Delivery

Concepts and practices

From startup to enterprise

Charles T. Betz

Minneapolis, Minnesota

Published by
Digital Management Academy, LLC
14 Sidney Place
Minneapolis, MN 55414

Digital Delivery
Concepts and Practices
First edition, LeanPub early release

Produced in the United States of America
Compiled on 2017/02/07 at 14:18:16

This PDF edition is meant to be read on a device. It has extensive internal and external linking. A print edition with page cross references and printed URLs is forthcoming. See the "Backlog and release notes" section in the back matter for more on this particular version.

Cover illustration by Go To Media, LLC

ISBN: 978-0-9981346-0-4

Publisher's note to readers:

Knowledge and best practice in this field change constantly. As new research and experience broaden our understanding, changes in methods or practices may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information or methods described herein. In using such information or methods they should be mindful of risks to themselves or others, including parties for whom they have a professional responsibility.

Although the author and publisher have made every effort to ensure that the information in this book was correct at press time, the author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

For information about special discounts for bulk purchases or for information on booking authors for an event, please visit www.dm-academy.com.

To my students, past, present, and future

Contents

List of Figures

List of Tables

Preface

I wrote my first book, *Architecture and Patterns for IT Service Management, Resource Planning and Governance: Making Shoes for the Cobbler's Children* in 2006, with a second edition in 2011. I presented the second edition at the national SEI Saturn conference in Minneapolis in 2013, where I was approached by Dr. Bhabani Misra, the head of the Graduate Programs in Software at the University of St. Thomas in St. Paul. Dr. Misra asked me to teach a class called IT Infrastructure Management (SEIS660), which was to cover not just technical topics but also process and governance.

The course (which has run every semester since January 2013) has been developed during an extraordinary period for IT and digital management. Even in 2013, the trend towards a new style of IT delivery, based on Agile and DevOps practices was notable and accelerating. At this writing, these approaches seem to have "crossed the chasm" in the words of Geoffrey Moore, and are becoming the dominant models for delivering information technology value. As this book describes, there are good reasons for this historical shift, and yet its speed and reach are still disorienting.

For three semesters I assigned my first book (*Architecture and Patterns for IT: Service Management, Resource Planning, and Governance*) as a required text for the class. However, I did not write this as a textbook, and its limitations became clear. While I gave considerable attention to Lean and Agile in writing the book, it has a strongly architectural approach, coming at the IT management problem as a series of views on a model. I do not recommend this as a pedagogical approach for a survey class. It also had a thoroughly enterprise perspective, and I began to question whether this was ideal for new students. Further thought led to the idea of the emergence model (detailed in the Introduction).

I proposed the idea of a third edition to my publisher — one that would pivot the existing material towards something more useful in class. They agreed to this and I started the rewrite. However, by the time I was halfway done with the first draft, I had a completely new book. Material from the previous work was more technical, and this book was more of a business analysis.

A number of factors converged:

- My view that the "medium is the message," which extends to the choice of authoring approach, toolchain, and publisher

- A desire to freely share at least a rough version of the book, both for marketing purposes and in the interests of giving back to the global IT community
- A desire to be able to rapidly update the book with as little friction as possible
- A practical realization that the book might get more uptake globally if it were available, at least in some form, as free and open source intellectual property
- The fact that I had already started to publish my labs on GitHub and had, in fact, developed a workable continuous delivery ("DevOps") toolchain (the Calavera project, which has attracted collaborators from the U.S., Spain, and Israel)

Ultimately, the idea of starting my own publishing company, and managing my own product, appeared both desirable and practical. The journey has been long and intense, taking easily twice as long as either of my first two books.

I have had several working titles, and am still debating the best. Comments appreciated.

- *Agile IT Management: From Startup to Enterprise*
- *Digital*
- *The Digital Professional: From Startup to Enterprise*
- *Digital Delivery: Concepts and Practices*

Many have assisted with this work:

Thanks to Dr. Bhabani Misra for asking me to teach at the University of St. Thomas and providing direction at key points.

Thanks to Stephen Fralippolippi and Roger K. Williams for being the first GitHub contributors.

Thanks to Jason Baker for text and technical collaboration.

Thanks to Mark Kennaley for guidance on open versus closed loop thinking.

Thanks to Glen Alleman for guidance on modern project management practices.

Thanks to Jeff Sussna for ongoing inspiration, Twitter feedback, discussion question ideas, and sourced quotes.

Thanks to Nicole Forsgren for links to articles on performance management.

Thanks to Evan Leybourn for detailed commentary on project management in chapter eight.

Thanks to Chris Little and Jabe Bloom for quote provenance.

Thanks to Lorin Hochstein for references.

Thanks to Gene Kim for ongoing mentoring and advice on writing and publishing and unwavering support and confidence in my efforts.

Thanks to Murray Cantor for key insights and graphic for Chapter 7.

Thanks to Rob England for ongoing discussion, significant input on the change/stability systems thinking, and inspiration and his work on Standard + Case.

Thanks to the faculty I have met and worked with on the Digital Curricula initiative for the Minnesota State System, including Firasat Khan, Mary Mosman, David Bahn, Amos Olagunju, Svetlana Gluhova, Mary Lebens, Justin Opatrny, Grant Spencer, Halbana Tarmizi.

Thanks to Go To Marketing Team (Will Goddard, Terry Brown, Francisco Piniero) for design assistance and invaluable partnership.

Thanks to Professor Pat Paulson for being the first adopter of the textbook, and thanks to his students for invaluable criticism and feedback.

Thanks to Majid Iqbal for significant input on change/stability systems thinking.

Introduction for Instructors and Trainers

Welcome to *Digital Delivery: Concepts and Practices*. So, what exactly IS this book?

- It is the first general, survey-level text on IT management with a specific Agile, Lean IT, and DevOps orientation.
- It has a unique and innovative learning progression based on the concept of organizational evolution and scaling.
- Because it is written with continuous integration and print-on-demand techniques, it can be continually updated to reflect current industry trends.

The book is intended for both the academic and industry training communities.

The book is intended for both the academic and industry training communities. There has been too much of a gap between academic theory and the day to day practices of managing digital products. Industry guidance has over the years become fragmented into many overlapping and sometimes conflicting bodies of knowledge, frameworks, and so forth. The emergence of Agile and DevOps as dominant delivery forms have thrown this already fractured ecosystem of industry guidance into chaos. Organizations and individuals with longstanding investments in guidance such as the IT Infrastructure Library (ITIL) [1] and the Project Management Body of Knowledge (PMBOK) are re-assessing these commitments. This book seeks to provide guidance for both new entrants into the digital workforce, as well as experienced practitioners seeking to update their understanding on how all the various themes and components of IT management fit together in the new world.

Digital investments are critical for modern organizations and the economy as a whole. Delivering them (defined as both creating and managing for value) can provide prosperity for both individuals and communities. Now is an ideal time to re-assess and synthesize the bodies of knowledge and developing industry consensus on how digital and IT professionals can and should approach their responsibilities.

The IT industry and the rise of digital

> Now agile methodologies — which involve new values, principles, practices, and benefits and are a radical alternative to command-and-control-

> style management — are spreading across a broad range of industries and functions and even into the C-suite. [213]
>
> — Darrell Rigby et al. *Harvard Business Review*

As an instructor, I ask you to consider the following two industry reports.

In September 2015, Minneapolis-based Target Corporation laid off 275 workers with IT skillsets such as business analysis and project management, while simultaneously hiring workers with newer "Agile" skills. As quoted by a local news site, Target stated:

"As a part of our transition to an Agile technology development and support model, we conducted a comprehensive review of our current structure and capabilities... we are eliminating approximately 275 positions and closing an additional 35 open positions. The majority of the impact was across our technology teams and was primarily focused on areas such business analysis and project management." [141]

Digital is a capability that needs to exist in every job.

Jim Fowler, Chief Information Officer at General Electric, says:

"When I am in business meetings, I hear people talk about digital as a function or a role. It is not. Digital is a capability that needs to exist in every job. Twenty years ago, we broke ecommerce out into its own organization, and today ecommerce is just a part of the way we work. That's where digital and IT are headed; IT will be no longer be a distinct function, it will just be the way we work. ... [W]e've moved to a flatter organizational model with "teams of teams" who are focused on outcomes. These are co-located groups of people who own a small, minimal viable product deliverable that they can produce in 90 days. The team focuses on one piece of work that they will own through its complete lifecycle... in [the "back office"] model, the CIO controls infrastructure, the network, storage, and makes the PCs run. The CIOs who choose to play that role will not be relevant for long." [115]

Modern management information systems (MIS) courses and textbooks, especially at the undergraduate, survey level, take an "outside-in" approach to the course material, seeking to orient **all** students (whether IT/MIS specialists or not) to the role and function of information systems and their possibilities and value in the modern enterprise. This book, by contrast, is an "inside-out" book intended to prepare the student for a career in digital industry. *Industry* is broadly defined as both those industries that offer digital products per se as well as industries that rely on digital technology instrumentally for delivering all kinds of products. A central theme of the book is that IT, considered as a component, represents an increasing proportion of **all** industrial products (both consumer and business-facing). This trend towards IT's increase is known as digital transformation.

Current MIS survey texts have some common characteristics:

- They tend to focus on the largest organizations and their applications of computing. This can lead to puzzling topic choices; for example, in one MIS text I reviewed, one of the first sections is dedicated to the problem of enterprise IT asset management — a narrow topic for the earlier sections of a survey course and increasingly irrelevant in the age of the cloud.

- Their learning progression (structure and narrative) is often arbitrary; for example, covering databases, networking, ERP systems, security, and so forth in various orderings.
- They do not (and this is a primary failing) cover Agile and its associated digital ecosystem at all well. Brief mentions of Agile may appear in sections on project management, but in general there is a lack of awareness of the essential role of Agile and related methods in accelerating digital transformation.
- Their coverage of cloud infrastructure can also be limited, even with new editions coming out every year. Topics like infrastructure as code go unaddressed.
- Finally, current texts often uncritically accept and cite "best practice" IT frameworks such as CMMI, ITIL, PMBOK, and COBIT. New digital organizations do not, in general, use such guidance, and there is much controversy in the industry as to the value and future of these frameworks. This book strives to provide a clear, detailed, and well-supported overview of these issues.

IT has had a reputation for expensive inflexibility.

IT, or the digital function, has had a history of being under-managed and poorly understood relative to peer functions in the enterprise. It struggles with a reputation for expensive inflexibility and Dilbert-esque dysfunction. The DevOps and Agile movements promise transformation but are encountering an entrenched legacy of

- enterprise architecture;
- program and project management;
- business process management;
- IT service management practices; and
- IT governance concerns.

Understanding and engaging with the challenges of this legacy are an ongoing theme throughout this introductory text. Some of the more radical voices in the sometimes give the impression that the legacy can be simply swept away. The following cautionary message from Mike Burrows shows that, in terms of the systems thinking at the core of Agile philosophy, this would be ill-advised:

"Some will tell you that when things are this bad, you throw it all away and start again. It's ironic: The same people who would champion incremental and evolutionary approaches to product development seem only too eager to recommend disruptive and revolutionary changes in people-based systems — in which the outcomes are so much less certain" ([43], loc. 827–829).

Changing complex systems should be done carefully.

IT management at scale within an organization is a complex system. The IT workforce, its collective experience, and its ongoing development (through education and training) is another complex system orders of magnitude larger. Complex systems do not respond well to dramatic perturbations. They are best changed incrementally, with careful monitoring of the consequences of each small change. (This is part of the systems theory foundation underlying the Agile movement.) This is why the book covers topics such as:

- Investment, sourcing, and people
- Project and process management
- Governance, risk, security, and compliance
- Enterprise information management
- Enterprise architecture and portfolio management

While these practices, and their associated approaches and policies, have caused friction with digital and Agile practitioners, they all have their reasons for existing. The goal of this book is to understand their interaction with the new digital approaches, but in order to do this we must first understand them on their own terms. It does no good to develop a critique based on misconceptions or exaggerations about what (for example) process management or governance is all about. Instead, we try to break these large and sometimes controversial topics into smaller, more specific topics — lowest common denominators, perhaps as follows:

- Work and effort
- Ordering of tasks
- Task dependencies
- Coordination
- Investment
- Cost of delay
- Planned versus unplanned work
- Estimation versus commitment
- Value stream versus skill alignment
- Repeatability
- Defined versus empirical process control
- Synchronization and cadence
- Resource demand
- Shared mental models
- Technical debt
- Risk

And so forth. By examining IT management in these more (clinical terms , we can develop a responsible critique of current industry best practices in content and form that will benefit students as they go out on their careers.

Note
A key choice in the book's evolution was to NOT include dedicated chapters on "Project Management" and "Process Management." Instead, more general chapter titles of "Coordination" and "Investment and Planning" were chosen. Rationale for the decision is given in those chapters and in Part III generally. Similarly, there is little coverage of "IT Service Management" per se; its significant concerns are seen throughout chapters 4-10.

A process of emergence

> Joseph Campbell popularized the notion of an archetypal journey that recurs in the mythologies and religions of cultures around the world. From Moses and the burning bush to Luke Skywalker meeting Obi wan Kenobi, the journey always begins with a hero who hears a calling to a quest.... The hero's journey is an apt way to think of startups. All new companies and new products begin with an almost mythological vision — a hope of what could be, with a goal few others can see.... Most entrepreneurs feel their journey is unique. Yet what Campbell perceived about the mythological hero's journey is true of startups as well: However dissimilar the stories may be in detail, their outline is always the same. [29]
>
> — Steve Blank *The Four Steps to Epiphany*

One of the most important and distinguishing features of this book is its emergence model. In keeping with the entrepreneurial spirit of works like Ries' *The Lean Startup*, the book adopts a progressive, evolutionary approach. The student's journey through it reflects a process of emergence. Such processes are often associated with founding and scaling a startup. There are many helpful books on this topic, such as the following:

- *Nail It Then Scale It* by Furr and Ahlstrom [96]
- *Scaling Up* by Harnish [111]
- *Startup CEO* by Blumberg [31]
- *The Lean Startup* by Ries [212]
- *Hello, Startup* by Brikman [33]

The emergence model and overall book structure is discussed in depth in the main introduction. Here, for the instructor, are some notes on the thought process.

The problem I set out to solve when I was first conceiving the book was the question of overall learning progression, or narrative.

The Stack and Lifecycle have been the basis for learning progressions in teaching technology.

I teach a required semester-long survey class on IT management at the University of St. Thomas-Minnesota, in the largest software engineering program in the country.

I had been trying to teach college students by walking them through architectural perspectives on the problem of IT management: the business, data, applications, and technical views many of us use regularly. Without going into detail, it wasn't working well and I started thinking about the problem. I noticed that there were two primary narratives or learning progressions being used to teach in computing:

- The "stack"
- The "lifecycle"

The stack is how the most rigorous topics are taught. Algebra is the foundation for trignometry, is the foundation for calculus, for example. Logic is needed for discrete math, required for automata and compilers and so forth. The stack is also how technology is described: physical, logical, and conceptual layers, for example. Architecture concepts are often stacks (conceptual/logical/physical, or business/application/data/technology, or the well known OSI network model.)

The systems lifecycle on the other hand, is how we tend to structure industry guidance. We plan and design, we build, we run. Guidance such as COBIT and ITIL show lifecycle influences, as do software engineering programs in colleges.

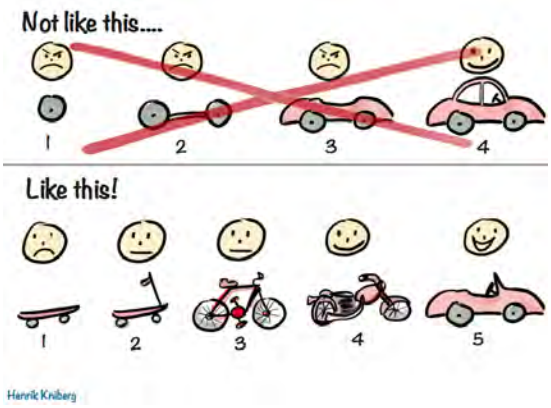


Figure 1: Systems evolve iteratively

However, both the stack and the lifecycle have limitations. The stack can fall into what venture capitalist Anshu Sharma calls the "stack fallacy," the "mistaken belief that it is trivial to build the layer above yours" [234]. It's also sometimes hard to know when you have covered the precursor material sufficiently. Finally, more foundational and theoretical topics can seem irrelevant to the student. ("In the beginning, the universe was created.") The lifecycle narrative is far too prone to promoting waterfall thinking, anathema to the current Agile and Lean Product Development approaches redefining digital industry.

Instead, the book's emergence narrative draws on systems theory, in particular John Gall's idea that "*A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system*" [97]. Henrik Kniberg created a compelling related visual image (Figure 1 [2]).

What would be the simplest possible thing that could work?

What if we treated the student's understanding as such a systems problem? What would be the simplest possible thing that could work? How would we iteratively

evolve their understanding, based on practical topics? Scaling seemed to be orthogonal to the other narratives (Figure 2).

As we'll cover in the main introduction, reading books on organizational scaling inspired the idea that growth does not happen smoothly; instead organizations tend to cluster at certain scales and struggle to grow to the next scale. Hence the overall structure of the book:

- Founder
- Team
- Team of Teams
- Enterprise

A key focus of the book is explaining what practices are formalized at which level of growth. The thought experiment is, "what would I turn my attention to next as my IT-based concerns scale up?" For example, I think work management (implying rudimentary workflow, e.g., Kanban) correctly comes before formalized project and/or process management, which in turn tend to emergence before enterprise governance practices (e.g., formalized risk management).

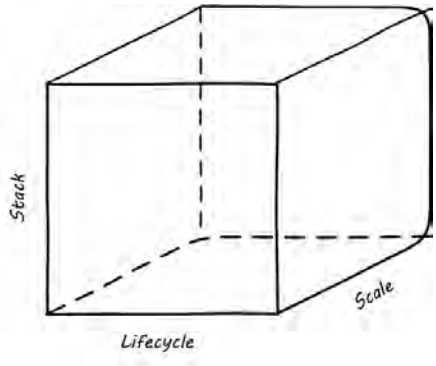


Figure 2: 3 narrative dimensions

Note that this would be a testable and falsifiable hypothesis if empirical research were done to inventory and characterize organization scaling patterns. If we found, for example, that a majority of organizations formalize governance, risk, security, and compliance practices before formalizing product management, that would indicate that those chapters should be re-ordered. In my experience, small/medium businesses may have formal product management but governance, risk, and compliance (GRC) are still *tacit*, not formalized. *This does not mean that GRC is not a concern*, but they have not yet instituted formal policy management, internal audit, or controls.

Product management is poorly addressed in most current college computing curricula.

The presence of product management at an early stage in the book (Chapter 4) is intended to provoke thought and debate. Product management is poorly addressed in most current college computing curricula as well as the reigning industry standards (e.g. TOGAF, PMBOK and ITIL). Yet formalizing it is one of the earliest concerns for a startup, and the imperatives of the product vision drive all that comes after. Evidence to this effect is seen (as of 2015) at the University of California at Berkeley I-School, which has replaced its Project Management course with Lean/Agile Product Management, taught currently by the esteemed Jez Humble, author of *Continuous Delivery, Lean Enterprise*, and co-author of *The DevOps Handbook*.

The book however is not a complete dismissal of older models of IT delivery. Wherever possible, new approaches are presented relative to what has gone before. The specifics of "what's different" are identified, in the interest of de-mystifying what can be fraught and quasi-religious topics. (Why is a Scrum standup or a Kanban board effective, in terms of human factors?)

The emergence model can also be understood as an individual's progression within a larger enterprise. Even if one starts from day one at a Fortune 100 corporation, I believe the progression of one's understanding still progresses from individual, to team, to "team of teams," to enterprise. Of course, one may cease evolving one's understanding at any of these stages, with corresponding implications for one's career.

Some of you may be familiar with the idea of a minimum viable product (MVP), minimum marketable release, or similar. In these terms, it is important to understand that each **section** of the book represents an MVP but not each chapter. One can't begin to deliver IT value without the components discussed in each of chapters one through three. The chapters of each section tend to be interdependent in other words.

This book does not cover specific technologies in any depth. Many examples are used, but they are carefully framed to not require previous expertise. This is about broader, longer lifecycle trends.

There is benefit to restricting the chapters to twelve, as a typical semester runs fourteen weeks, and the book then fits well, with one chapter per class and allowing for an introductory session and final exam. Of course, a two-semester series, with two weeks per chapter, would also work well. Each half of the book is also a logical unit. I have spent considerable time thinking (agonizing) about the correct ordering of the chapters within these sections. This is possibly the tenth or twelfth version of the chapter ordering. This is an area where I want critical review but also have strong opinions.

Labs

With three chapters in each section, the book can be covered in one intense semester at a chapter a week, although expanding it to a two-semester treatment would allow for more in-depth coverage and increased lab exposure. I give great credit to both my first cohort of students and Dr. Bhabani Misra for challenging me to add a practical component to the course. This required new thinking on my part. How to demonstrate IT management at scale in a lab setting? I have learned that a hands-on component is essential, as IT management discussions can be abstract and meaningless to many students. ("Incidents are different from problems!")

Ten years ago, the best that would have been possible would be paper case studies, perhaps augmented with spreadsheets. But new options are now available. The power of modern computers (even lightweight laptops) coupled with the widespread availability of open source software makes it is now possible to expose students to industrial computing in a meaningful, experiential way. I have found great utility in the use of lightweight virtualization technologies such as Vagrant, VirtualBox, and

Docker. I recommend this approach wholeheartedly. I am always interested in hearing from other instructors who are working from the same approach. At the time of this writing, I maintain my labs (table Table 1) publicly on GitHub. My syllabus and lab structure are under continual improvement.

Lecture	**Topic**	**Business lab**	**Technical lab**	**Team size**
Course introduction	Structure, approach	None	SSH & workstation setup	Individual
Part I: FOUNDER				
Chapter 1	IT value	Defining an IT product — review SaaS examples	Linux command line	2
Chapter 2	IT infrastructure	Reviewing current SaaS offerings	Cloud and infrastructure as code — configuring a Vagrant machine manually & w/ script that is checked in to GitHub & modified	2
Chapter 3	Applications		Continuous delivery pipeline	2
Part II: TEAM				
Chapter 4	Product management	User stories; fail fast/risk mgmt	Behavior-driven development	6–8
Chapter 5	Work management	Scrum, ticketing, and Kanban		6–8
Chapter 6	Operations management	Service definition	Monitoring (Calavera + Nagios)	6–8
Part III: TEAM OF TEAMS				
Chapter 7	Coordination	Organizational forms & communication channels (paper exercise?)	iTOP ITSM suite	¿ 11 (full class)

Lecture	Topic	Business lab	Technical lab	Team size
Chapter 8	Investment and planning		Architecture game	¿ 11 (full class)
Chapter 9	Organization and culture		Game or paper exercise	¿ 11 (full class)
Part IV: ENTERPRISE				
Chapter 10	Security, governance, risk, and compliance		Scanning VMs for vulnerabities with Lynis	5
Chapter 11	Enterprise information management	Data and records management exercises		5
Chapter 12	Architecture and portfolio	Portfolio investment simulation exercise		5

Table 1: Course Labs

I use a central server in teaching my classes, but even that is not necessary. This class can be taught with a zero computing budget, assuming that each team of students at least has access to a modern laptop (recommend 8 gigabytes of RAM and 1 terabyte drive) and a fast Internet connection. As of this writing, I am using free and open source versions of Chef, Jenkins, iTOP, jUnit, Ant, and other tools (see GitHub for the current approach).

Some may question the inclusion of command-line experience, but without some common technical platform, it is hard to provide a meaningful, hands-on experience in the first half of the course. I structure my class on the assumption that the students are at least willing to learn computing techniques, with no prerequisites beyond that. Not even a programming language is required; the Java currently used as a sample is minimal.

Truly beginning students will have to work at the Linux tutorials, but all they need master is basic command line navigation, and I have found this possible with a diverse student body, some with no previous direct experience. The labs for the second half of the course use games, experiential paper-based classroom exercises, GUI-based software, databases, and office productivity tools.

``

Introduction

For the student

This is a *survey* text, intended for the advanced undergraduate or graduate student interested in the general field of applied IT management. It is also intended for the mid-career professional seeking to update their understanding of IT management's evolution, especially in light of the impact of Agile and DevOps.

The book is grounded in basic computing fundamentals but *does not require any particular technical skills to understand*. You do **not** need to have taken any courses in networking, security, or specific programming languages to understand this book. However, you occasionally will be presented with light material on such topics, including fragments of programming languages and pseudocode, and you will need to be willing to invest the time and effort to understand.

This book makes frequent reference to digital startups — early stage companies bringing new products to market that are primarily delivered as some form of computer-based service. Whether or not you intend to pursue such endeavors, *the startup journey is a powerful frame for your learning*. Large information technology organizations in enterprises sometimes gain a reputation for losing sight of business value. IT seems to be acquired and operated for its own sake. Statements like "we need to align IT with the business!" are too often heard.

A digital startup exposes with great clarity the linkage between IT and "the business."

A digital startup exposes with great clarity the linkage between IT and "the business." The success or failure of the company itself depends on the adept and responsive creation and deployment of the software-based systems. Market revenues arrive, or do not, based on digital product strategy and the priorities chosen. Features the market doesn't need? You won't have the money to stay in business. Great features, but your product is unstable and unreliable? Your customers will go to the competition.

The lessons that digital entrepreneurs have learned through this trial by fire shed great light on IT's value to the business. Thinking about a startup allows us to consider the most fundamental principles as a sort of microcosm, a small laboratory model of the same problems that the largest enterprises face.

Verne Harnish, in the book *Scaling Up* ([111], pp. 25-26), describes how companies tend to cluster at certain levels of scale. (See Figure 3 [3] (p. 25).)] The majority of firms never grow beyond a founder; a small percentage emerge as a viable team of 8-12, and even smaller numbers make it to the stable plateaus of 40-70 and 350-500. The "scaling crisis" is the challenge of moving from one major level to the next.

(Harnish uses the more poetic term "Valley of Death.") This scaling model, and the needs that emerge as companies grow through these different stages, is the basis for this book's learning progression.

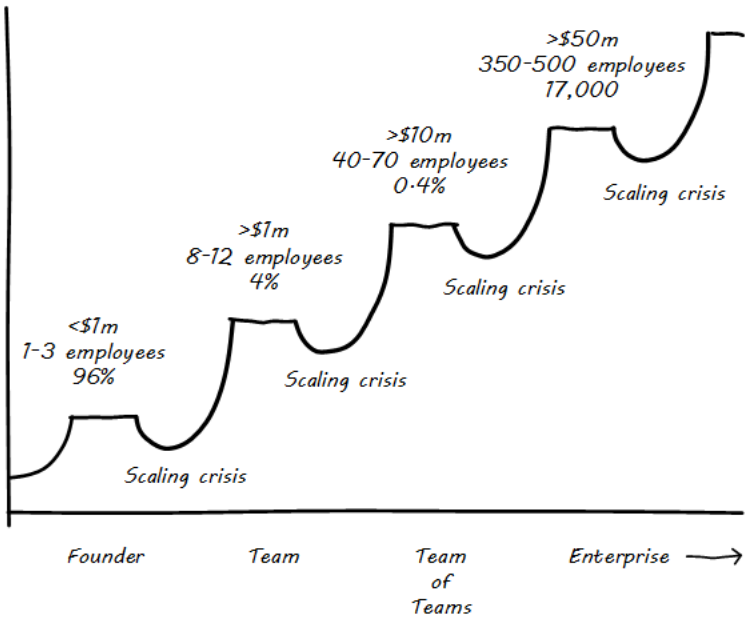


Figure 3: Organizations cluster at certain sizes

However, this is not a textbook (or course) on entrepreneurship. It remains IT-centric. **And, the book is also intended to be relevant to students entering directly into large, established enterprises.** In fact, it prepares the student for working in all stages of growth because it progresses through these four contexts:

- Individual (founder)
- Team
- Small company (team of teams)
- Enterprise

Whether in a startup or on a journey within a larger, established organization, you will (hopefully) become aware as you progress through a broadening context:

- Other team members
- Customers
- Suppliers
- Sponsors

- Necessary non-IT capabilities (finance, legal, HR, sales, marketing, etc.)
- Channel partners
- Senior executives and funders
- Auditors and regulators

Part of maturing in one's career is understanding how all these relationships figure into your own overall system of value delivery. This will be a lifelong journey for the student; the author's intent is to provide some useful tools.

This book's structure

Figure 4: IT management evolutionary model (read bottom to top)

In Figure 4 is a conceptual illustration of an IT management progression (read the figure bottom to top). Elaborating the outline into chapters, we have:

How do we define what we are building and quickly validate our ideas?

I. **Founder**

1. **IT value.** Why do we need computers? What can they do for us?
2. **IT infrastructure.** We want to build something. We have to choose a platform first.
3. **IT applications.** Let's start building something of use to someone.

II. **Team**

1. **Product management.** What exactly is it that we are building? What is the process of discovering our customer's needs and quickly testing how to meet them? How do we better define the product vision, and the way of working towards it, for a bigger team?

2. **Work management.** How do we keep track of what we are doing and communicate our progress and needs at the simplest level?
3. **Operations management.** How do we sustain this surprisingly fragile digital service in its ongoing delivery of value?

III. **Team of Teams**

1. **Coordination.** When we have more than one team, they need to *coordinate*, which we define as "the process of managing dependencies among activities." There are many synchronization techniques to help us coordinate, including project and process management and Agile concepts. What is the future of process management as a delivery model?
2. **Investment and planning.** We make investments in various products, programs, and/or projects, and we are now big enough that we have portfolios of them. How do we decide? How do we choose and work with our suppliers? How do we manage the finances of complex digital organizations? What is the future of project management as a delivery model?
3. **Organization and culture.** We're getting big. How do we deal with this? How are we structured, and why that way? How can we benefit from increasing maturity and specialization while still maintaining a responsive digital product? How do we hire great people and get the most out of them? What are the unwritten values and norms in our company and how can we change them?

IV. **Enterprise**

1. **Governance, risk, security, and compliance.** We need to cope with structural and external forces investors, directors, regulators, vendor partners , security adversaries, auditors to whom we are ultimately accountable or who are otherwise defining our options. What are their motivations? How do we understand and control risk? How are we assured that our strategy, tactics, and operations are reasonable, sound, and thorough? And how do we protect ourselves from adversaries?
2. **Enterprise information management.** We've been concerned with data, information, and knowledge since the earliest days of our journey. But at this scale, we have to formalize our approaches and understandings; without that, we will never capture the full value available with modern analytics and big data. Compliance issues are also compelling us to formalize here.
3. **Architecture and portfolio.** We need to understand the big picture of interacting lifecycles, reduce technical debt and redundancy, accelerate development through establishing platforms, and obtain better economies of scale. We do so in part through applying techniques such as visualization, standardization, and portfolio management. We need to define our investment strategy based on a sound understanding of both business needs and technology limitations.

V. **Appendices**

1. **The major frameworks**
2. **Project management**
3. **Process modeling**
4. **References**
5. **Glossary**
6. **Backlog**
7. **Colophon**
8. **Author biography**

Warning
The boundary between the "Team" and the "Team of Teams" is a challenging area, and industry responses remain incomplete and evolving.

Emergence means formalization

The emergence model seeks to define a likely order in which concerns are **formalized**. Any concern may of course arise at any time—the startup founder certainly is concerned with security! Formalization means at least one or more of the following:

- Dedicated resources
- Dedicated organization
- Defined policies and processes
- Automated tooling

In my experience, for example, startups avoid formalized process and project management . To the extent the concerns exist, they are *tacit* (understood or implied; suggested; implicit). Certainly, a small startup does not invest in an enterprise-class service desk tool supporting a full array of IT management processes or a full-blown project management office with its own vice president and associated portfolio automation. Simple work management, with a manual or automated Kanban board, is likely their choice for work management.

Specialization and formalized processes emerge as an organization scales.

But by the time they are a team of teams, specialization has emerged and more robust processes and tools are required. Finally, the more complex, enterprise-scale concerns at the end of the book are presented as part of a logical progression.

The danger of course is that the formalization effort may be driven by its own logic and start to lose track of the all-critical business context. By carefully examining these

stages of maturation, and the industry responses to them, it is the author's hope that the student will have effective tools to critically engage with the problem of scaling the digital organization.

Finally, the scaling model also emphasizes the critical importance for the reader of the high-performing, multi-skilled, collaborative team. Coordination and enterprise problems must be given their due, but too often the proposed solutions destroy the all important team value. As stated elsewhere in this book, it is possible that there is no higher unit of value in the modern economy than the high-performing team. Maintaining the cohesion and value of this critical asset is presented as a clear priority throughout the subsequent chapters.

Assumptions about the reader

- This book is written at the advanced undergraduate/graduate student level. It is also intended for mid-career and senior IT practitioners seeking to update their knowledge.
- It is currently available only in English.
- There is no assumption of deep IT experience, but it is assumed that the person will interact with computers in some capacity and has basic technical literacy. They should, for example, understand the concept of an operating system. An A+ certification, or an intro to networking or programming class for example, would more than adequately prepare someone for this book.
- A person completely unfamiliar with computing will need to supplement their reading as suggested throughout the text. There is a wealth of free and accurate information on IT fundamentals (e.g., computing, storage, networking, programming, etc.), and this book seeks more to curate than replicate.

Notes

[1] Technically, ITIL is now just the abbreviation, but we spell it out here for reference.

[2] Image credit Henrik Kniberg http://blog.crisp.se/author/henrikkniberg, permission pending

[3] similar to [111]

Part I

Founder

This is the introduction to part I. In this section, we explore the fundamentals of information technology delivery.

Scenario

You are working in a startup, alone or with one or two partners. You are always in the same room, easily able to carry on a running conversation about your efforts and progress. You have no time or resources to spend on anything except keeping your new system alive and running.

Chapter 1: IT Value

Chapter 1 introduces you to the fundamental concepts of IT value that serve as a basis for the rest of the course. Why do people want computing (IT) services? What are the general outlines of their structure? How do they come into being? How are they changed over time?

All of this is essential to understand for your scenario; you need to understand what computers can do and how they are generally used if you are going to create a product based on them.

This chapter also covers the basics of how you'll approach building a product. It's assumed you won't develop an intricate, long-range plan but rather will be experimenting with various ideas and looking for fast feedback on their success or failure.

Chapter 2: IT Infrastructure

In this chapter, you have a general idea for a product and are ready to start building it. But not so fast... you need to decide some fundamentals first. How will your new product run? What will you use to build it?

It's not possible to begin construction until you decide on your tools.

It's not possible to begin construction until you decide on your tools. This chapter will provide you an overview of computing infrastructure including cloud hosting and various approaches to system configuration.

This chapter also presents an overview of source control, as even your infrastructure depends on it in the new world of "infrastructure as code."

Chapter 3: Application delivery

Finally, you're ready to start building something. While this is not a book on software development or programming languages, it's important to understand some basics and at least see them in action.

This is also where we introduce the concept of "DevOps"; it's not just about writing code but about the entire end-to-end system that gets the code you are writing from your workstation, into collaborative environments, and finally to a state where it can be accessed by end users. From source repository to build manager to package repository to production, we'll cover a basic toolchain that will help you understand modern industrial practices.

This section's lab approach

While this is not a book about any particular computing language or platform, we need to describe some technical fundamentals. We'll do so in as neutral a manner

as possible. However, this book's accompanying labs are based on Ubuntu Linux and git, the distributed version control system created by Linus Torvalds to facilitate Linux development.

Important

Part II, like the other parts, needs to be understood as a unified whole. In reality, digital entrepreneurs struggle with the issues in all three chapters simultaneously.

Chapter 1

IT Value

Introduction

You are now a producer (or soon to be) of IT services.

As noted at the outset, you are a small core of a startup. Your motivations are entrepreneurial ; you want to create a successful business. You might be housed within a larger enterprise, but the thought experiment here is that you have substantial autonomy to order your efforts. You want to do something that has a unique digital component. Regardless of your business, you will need accounting and legal services at a minimum and, very quickly, payroll and HR and so forth. Those things can (and should) be purchased as commodity services if you are a small entrepreneur (I am not aware of any convincing arguments to the contrary, unless you are absolutely on the smallest of shoestring budgets and can work 100-hour weeks). Your unique value proposition will be expressed to some degree in unique IT software. While this software may be based on well-understood products, the configuration and logic you construct will be all your own. Because of this, you are now a producer (or soon to be) of IT services.

Before we can talk about building and managing information technology (IT), we need to understand what it is and why people want it. We'll start this chapter by looking at an IT value experience that may seem very familiar. Then we'll dig further into concepts like the IT stack and the IT service and how they change over time.

Chapter outline

- An IT value experience
- What is information technology?
- The IT *service* and the IT *stack*
- The IT service
- IT changing over time
- The digital context

- Conclusion

Learning objectives for this chapter

- Explain "IT value" in everyday terms
- Distinguish between IT service and IT system
- Discuss how IT services change over time
- Describe various ways of understanding the context in which digital systems are developed and digital value is delivered.

What is IT value?

An IT value scenario

Figure 1.1: Dinner out tonight?

Consider the following scenario:

A woman (Figure 1.1 [4]) is wondering if she can afford to dine out that evening. She uses her mobile device to access her banking information and determines that in fact she does have enough money to do so. She also uses her mobile device to make a reservation and contact some friends to join her. Finally, she uses social navigation software to avoid heavy traffic, arriving at the restaurant in time for an enjoyable evening with her friends.

Information technology pervaded this experience. The origins, layers, and complex connections of the distributed systems involved are awe-inspiring to consider.

Important

Don't worry about the technological terms for now. This is an introductory text. You may see terms below that are unfamiliar (model-view-controller, IP, packet switching). If you are reading this online, you can follow the links, but it's not required. As you progress in your career, you will always be encountering new terminology. Part of what you need to learn is when it's important to dig into it and when you can let it pass for a time. You should be able to understand the gist presented below that these are complex systems based on a wide variety of technologies, some of them old, some new.

The screen on her cell phone represents information accessed and presented via a model-view-controller framework, implemented in the latest version of JavaScript, running on an interpreter that would have taxed a mainframe thirty years ago. The communication with her bank's central systems is supported by 4G LTE data which in turn relies on the high-volume IP backbone networks operated by the telecommunications carriers, based on research into packet switching now approaching fifty years old. The application operating on the cell phone interacts with core banking systems via sophisticated and highly secure middleware, crossing multiple network control points. This middleware talks in turn to the customer demand deposit system that still runs on the mainframe.

Mainframe software may date back decades, and still run well.

The mainframe is now running the latest version of IBM's z/OS operating system (a direct descendant of OS/360, one of the most significant operating systems in the history of computing). The customer demand deposit banking application running on the mainframe is still based on code written in the lowest level assembler. Some of the comments in this code date back to the 1970s. It has been tuned and optimized over the decades into a system of remarkable speed and efficiency. Although replatforming it is periodically discussed, the cost/benefit ratio for such a project has to date not been favorable.

Figure 1.2: Digital made this gathering easier

The reservation system looks similar on the mobile device, but the network routes it to a large cloud data center hosting the reservation system. The back end application here is very different from the banking system; the programming languages are newer, the database is structured very differently, and the operating system is Linux.

Finally, the navigation software looks much like the reservation system, as it too is based on the cloud. However, the system is much more active as it is continually processing inputs from millions of drivers in thousands of cities and updating traffic maps for those drivers in real time so that they can choose the most optimal route to their destinations (e.g., dinner). The capabilities of this system are comparable to an air traffic control system, and yet it is available as a free download for our IT user.

The resulting value (as in Figure 1.2 [5]) is clear:

- In an earlier era, our user might have stayed in for fear of bouncing a check, or she might have gone out and dined beyond her means.
- The phone line at the restaurant might have been busy, so she might have risked showing up with no reservation.

- Before texting and social media, she might not have been able to reach her friends as easily.
- Without the traffic application, she might have run into a huge midtown traffic jam and been half an hour late.

Instructor's note: This case is intended to reflect current service thinking, e.g. [263], [246].

Clearly, information technology added value to her life and helped maximize her experience of social enjoyment.

Various forms of IT value

Digital technology generates value in both direct and indirect ways.

As we have seen, there are many ways in which digital systems deliver value . Some systems serve as the modern equivalent of file cabinets: massive and secure storage for financial transactions, insurance records, medical records, and the like. Other systems enable the transmission of information around the globe, whether as emails, web pages, voice calls, video on demand, or data to be displayed in a smartphone application (app). Some of these systems support engaged online communities and social interactions with conversations, media sharing, and even massive online gaming ecosystems. Yet other systems enable penetrating analysis and insight by examining the volumes of data contained in the first two kinds of systems for patterns and trends. Sophisticated statistical techniques and cutting-edge approaches like neural network-based machine learning increase the insights our digital systems are capable of, at a seemingly exponential rate.

Digital technology generates value in both direct and indirect ways. People have long consumed (and paid for) communication services, such as telephone services. Broadcast entertainment was a different proposition, however. The consumer (the person with the radio or television) was not the customer (the person paying for the programming to go out over the airwaves). New business models sprung up to support the new media through the sale of advertising air time. In other words, the value proposition was indirect, or at least took multiple parties to achieve: the listener, the broadcaster, and the advertiser. Finally, some of the best known uses of digital technology were and are very indirect — for example, banks and insurance agencies using the earliest computers to automate the work of thousands of typists and file clerks.

From these early business models have evolved and blossomed myriads of creative applications of digital technology for the benefit of human beings in their ongoing pursuit of happiness and security. We see the applications mentioned at the outset: online banking, messaging, restaurant reservations, and traffic systems. Beyond that we see the use of digital technology in nearly every aspect of life. (And I say "nearly" only because I am a cautious person.)

Digital technology increasingly pervades every industry.

Digital and information technology pervades all of the major industry verticals (e.g., manufacturing, agriculture, finance, retail, healthcare, transportation, services) and common industry functions (e.g., supply chain, human resources, corporate finance, and even IT itself). Digital systems and technologies also are critical components of larger scale industrial, military, and aerospace systems. For better or worse, general

purpose computers are increasingly found controlling safety-critical infrastructure and serving as an intermediating layer between human actions and machine response. Robotic systems are based on software, and the Internet of Things ultimately will span billions of sensors and controllers in interconnected webs monitoring and adjusting all forms of complex operations across the planet.

Defining Information Technology

What is IT, anyways?

We've started this book in the previous section by providing an example of digital or IT value, without much discussion of how it is delivered. This is deliberate. But what is IT (Information Technology), anyways?

- The computers? The networks?
- The people who run them?
- That organization under a Chief Information Officer that loves to say "no" and is always slow and expensive?

None of these are how this book defines "IT." Although this is not a technical book on computer science or software engineering, the intent is that it reflects and is compatible with foundational principles.

"Information technology" is ultimately based on the work of Claude Shannon, Alan Turing, Alonzo Church, John von Neumann, and the other pioneers who defined the central problems of information theory, digital logic, computability, and computer architecture.

Additionally, as an organizational function, information technology also draws on organizational theory, systems theory, human factors and psychology, and more recent concepts such as design thinking, among many other areas. Discussions of "information technology" become contentious because some think of the traditional organization, while others think of the general problem area. IT has a long history as a corporate function, a single hierarchy under a powerful Chief Information Officer. This model has had its dysfunctions, including a longstanding reputation for being slow and expensive. Often, when one encounters the term "IT," the author using the term is referring to this organizational tradition.

We are less interested in the future of IT as a distinct organizational structure. There are many different models, from fully centralized to fully embedded. Organizational structure will be discussed in Part III.

Digital management uses many different organizational structures.

For this book, we define "Information technology" in terms of its historic origins. We look to IT's common origins in automating the laborious and error prone processes of computation, through the application of digital logic technologies based on information transmission.

Regardless of organizational form or delivery methods, IT is defined by these origins. It does not matter if the application developers and systems engineers ultimately report up through the CIO, the CMO, the CFO, or the COO. There are common themes throughout IT and digital as a professional domain: the fragility and complexity of these systems, the need for layered abstractions in their management, and more.

IT and digital transformation

> IT doesn't matter. [51]
>
> — Nicholas Carr

> Software is eating the world. [13]
>
> — Mark Andreessen

> The digital realm is infusing the physical realm, like tea in hot water. [251]
>
> — Jeff Sussna *Designing Delivery*

IT increasingly permeates business operations and social interactions. The breadth and depth of IT support for virtually all domains of society continues to expand. Lately, this is known as digital transformation [269].

The role of information technology seems critical to society and the economy, but there are various points of view. Nicholas Carr, in his controversial *Harvard Business Review* article "IT Doesn't Matter," recognized that IT was becoming commoditized in an important sense [51]. As Cloud providers started to offer utility-style computing, the choice of particular vendors of computers was no longer strategic. Looking to history, Carr argued that just as businesses no longer have "Vice Presidents for Electricity," so businesses no longer need Chief Information Officers or dedicated IT departments.

Note

A "commodity" product is one that is offered from a variety of suppliers, with little or no difference between their offerings. Commodity products tend to compete on price, not on differences in features. Wheat is a commodity. Sports cars are not. "Commoditization" is the process by which products that used to compete by being different, increasingly compete on price.

IT will never be a commodity like electrical power.

Carr has insight — there is no question IT is becoming pervasive — but he ultimately reflects a narrow view of what "IT" is. If "IT" were merely computation at the lowest level — just shuffling bits of information around, doing a little math — then perhaps it could be embedded throughout a business like electricity.

But IT has emergent aspects that are not comparable to electrical power. As it pervades all dimensions of business operations, it brings its concerns with it: complexity, fragility, and the skills required to cope with them.

One watt of electrical power is like any other watt of electrical power, and can usefully be seen as a commodity. We can use it to run toasters, hair dryers, or industrial paint mixers, and there is little concern (beyond supply and demand management) that the consumption of power by the paint mixer will affect the toaster. It's also true that one cycle of computing, in a certain sense, is like any other cycle. But information technology systems interact with each other in surprising and unpredictable ways, orders of magnitude more complex than electrical power grids. (This is not to imply the modern electrical grid is a simple system!)

IT also radically transforms industries: from retail to transportation to manufacturing to genetics. Applied software-centric IT is unleashing remarkable economic disruption.

A lawyer may depend on a cell phone, and (in keeping with Carr) beyond its provision as a commodity service, needs little else to deliver the legal strategies a firm needs. A graphic designer may use computerized graphic tools, but these have become relatively standardized and commoditized in the past twenty years, and probably are not a source of competitive advantage in the quest for new marketing clients.

On the other hand, consider a text analytic algorithm that replaces thousands of paralegals, resulting in order-of-magnitude more accurate legal research in a fraction of cost and time. This **is** strategic and disruptive to the legal community. A superior supply chain algorithm, and the ability to improve it on an ongoing basis, may indeed elevate a logistics firm's performance above competitors. In cases like these — and they seem to be increasing — IT matters very much. The annual State of DevOps research finds that "Firms with high-performing IT organizations were twice as likely to exceed their profitability, market share, and productivity goals." [89]

In the digitally transforming economy, traditional "back office" IT organizations find themselves called on to envision, develop, and support market-facing applications of IT. And what starts with one market-facing use case can quickly expand into entire portfolios. It is such cases that are of particular concern in this book. Ultimately, it is possible that IT is the **most** strategic capability an organization can invest in. As Diomidis Spinellis, editor in chief of *IEEE Software* notes [245],

"other industries are also producing what's in effect software (executable knowledge) but not treating it as such … Although many industries have developed their own highly effective processes over the years, software engineering maintains an essential advantage. It has developed methods and tools that let even small teams manage extremely high complexity … This advantage is important because the complexity in non-software activities is also increasing inexorably … [T]he time has come to transform our world … by giving back to science and technology the knowledge software engineering has produced."

This ability to manage complexity, to turn tacit into explicit and formalize the previously unstructured, is an essential aspect of digital transformation.

Defining "IT"

So, how do we define an IT problem, as opposed to other kinds of business problems? An IT problem is any problem where you are primarily constrained by your capability

and understanding of IT.

- If you need computer scientists or engineers who understand the fundamentals of information theory and computer science, you are doing IT.

- If you need people who understand when your information-centric problems might need to be referred to such theorists and engineers, you are likely doing IT.

- If you need people who are skilled in building upon those fundamentals, and operating technical platforms derived from them (such as programming languages, general purpose computers, and network routers), you are doing IT.

Regardless of whether IT is housed under a traditional CIO, an operations capability, a Chief Marketing Officer, or a "line of business", when it is critical to operations certain concerns inevitably follow:

- Requirements (i.e. your intent for IT)

- Sourcing and provisioning

- IT-centric product design and construction

- Configuration and change management

- Support support

- Improvement

Newcomers who propose changes to these practices in hopes of making IT more "agile" are often surprised to find that these concerns were not mere bureaucracy, but instead had well grounded origins in past failures. Ignoring these lessons is perilous. And yet, the traditional, process-heavy IT organization does seem dysfunctional from a business point of view: a central theme of this book.

IT services, systems, and applications

Inside an IT service

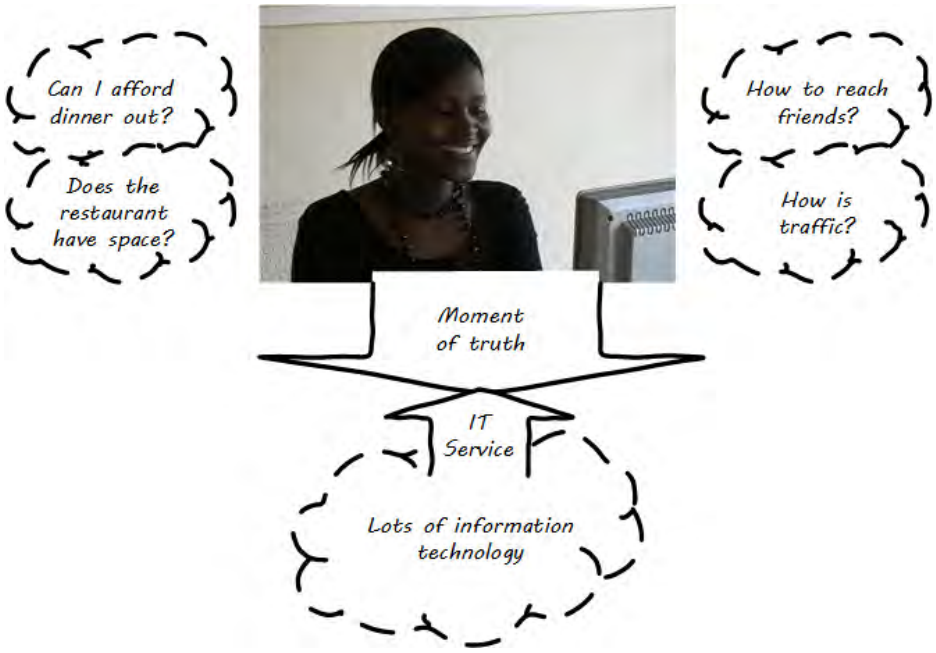


Figure 1.3: The basis of IT value

Let's examine our diner's value experience (Figure 1.3 [6]) in more detail, without getting unnecessarily technical, and clarify some definitions along the way. The first idea we need to cover is the "moment of truth." In terms of information technnology, this English-language cliche represents the user's experience of value.

In the example, our friend seeking a relaxing night out had several moments of truth:

- Consulting her bank balance, and subsequent financial transactions also reflecting what was stated to her
- Making a reservation and having it honored on arrival at the restaurant
- Arriving on time to the restaurant, courtesy of the traffic application
- And most importantly, having a relaxed and refreshing time with her friends.

Each of these individual value experiences was co-created by our friend's desire for value, and the response of a set of IT resources.

Important
The "moment of truth" represents the user's experience of value, from a product, good, or service.

In order to view her balance, our user is probably using an application downloaded from a "store" of applications made available to her device . On her device, this "app" is part of an intricate set of components performing functions such as:

- accepting "input" (user intent) through a screen or voice input
- processing that input through software and acting on her desire to see her bank balance
- connecting to the phone network
- securely connecting over the mobile carrier network to the Internet and then to the bank
- identifying the user to the bank's systems
- requesting the necessary information (in this case, an account balance)
- receiving that information and converting it to a form that can be represented on a screeen
- finally, displaying the information on the screen

The application, or "app," downloaded to the phone plays a primary role, but is enabled by:

- the phone's operating system and associated services
- the phone's hardware
- the telecommunications infrastructure (cell phone towers, long distance fiber optic cables, switching offices, and much more)

Of course, without the banking systems on the other end, there is no bank balance to transmit. These systems are similar, but on a much larger scale than our friend's device:

- Internet and middleware services to receive the request from the international network
- Application services to validate the user's identity and route the request to the appropriate handling service
- Data services to store the user's banking information (account identity and transactions) along with millions of other customers
- Many additional services to detect fraud and security attacks, report on utilization, identify any errors in the systems, and much more.
- Physical data centers full of computers and associated hardware including massive power and cooling infrastructure, and protected by security systems and personnel.

Consider: what does all this mean to our user? Does she care about cell phone towers, or middleware, or triply-redundant industrial-strength Power Distribution Units ? Usually, not in the least.

Therefore, as we study this world, we need to maintain awareness of her perspective. Our friend is seeking some value that IT uniquely can enable, but does not want to consider all the complexity that goes into it. She just wants to go out with friends. The moment of truth (Figure 1.4) depends on the service; the service may contain great complexity, but part of its success lies in shielding the user from that complexity.

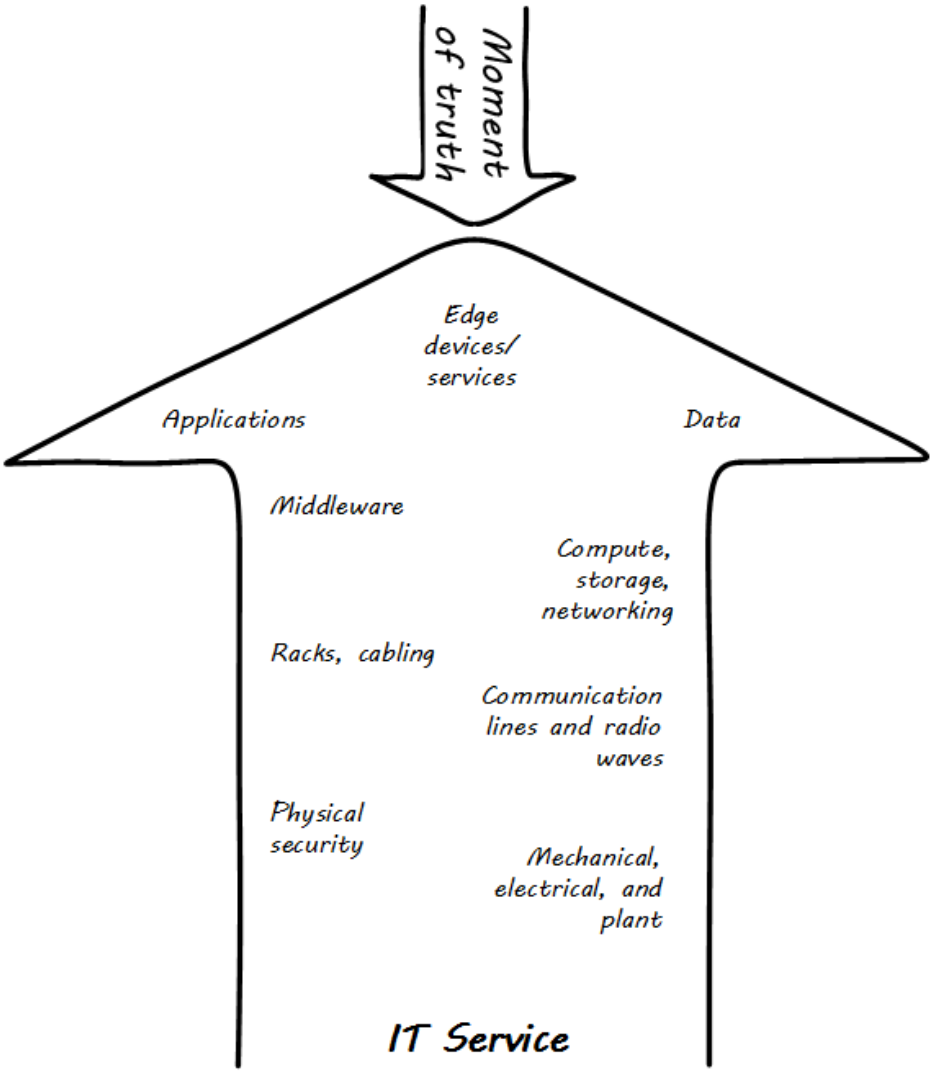


Figure 1.4: The IT stack supports the moment of truth

Chapter 2

Infrastructure Management

Introduction

As mentioned in the Part Introduction, you cannot start developing a product until you decide what you will build it with. (You may have a difficult time writing an app for a mobile phone if you choose the COBOL programming language!) You also need to understand something of how computers are operated, enough so that you can make decisions on how your system will run. Most startups choose to run IT services on infrastructure owned by a Cloud provider, but there are other options. Certainly, as you scale up, you'll need to be more and more sophisticated in your understanding of your underlying IT services.

Figure 2.1: Racks in a data center

Configuring your base platform is one of the most important capabilities you will need to develop. You'll never stop doing it. The basis of modern configuration management is version control, which we cover here.

This is one of the more technical chapters. Supplementary reading may be required for those completely unfamiliar with computing. See Assumptions of the Reader for notes on the book's approach. [11]

Chapter summary

- Introduction
  - Chapter summary
  - Learning objectives

- Infrastructure overview
  - What is Infrastructure?
  - Basic IT infrastructure concepts
- Choosing infrastructure
  - From "physical" compute to Cloud
  - Virtualization
  - Why is virtualization important?
  - Virtualization versus cloud
  - Containers and looking ahead
- Infrastructure as code
  - A simple infrastructure as code example
- Configuration management: the basics
  - What is version control?
  - Package management
  - Deployment management
- Topics in IT infrastructure
  - Configuration management, version control, and metadata
- Conclusion
  - Discussion questions
  - Research & practice
  - Further reading

Learning objectives

- Understand fundamental principles of operating computers as infrastructure for a service
- Understand Cloud as a computing option
- Understand basic principles of "infrastructure as code"
- Understand the importance and basic practices of version control and why it applies to infrastructure management

Infrastructure overview

In the previous chapter, you were introduced to the concept of a Moment of Truth , and in the final exercises asked to think of a product idea. Some part of that product requires writing software, or at least configuring some IT-centric system. (IT being defined as in Chapter 1.) You presumably have some resources (time and money). It's Monday morning, you have cleared all distractions, shut down your Twitter and Facebook feeds, and are ready to start building.

Not so fast.

Before you can start writing code, you need some kind of a platform. It's hard to build before you decide on your tools. You need to decide what language programming language you are going to write in, or what framework you are going to configure, and how that effort is going to result in an operational system capable of rendering IT services. You are probably swimming in a sea of advice and options regarding your technical choices. In previous decades, books such as this might have gone into the specifics of particular platforms: mainframe vs. minicomputers, COBOL vs FORTRAN, Windows vs Unix, etc.

At this writing, JavaScript is a leading choice of programming languge, in conjunction with various frameworks and NoSQL options (e.g. the MEAN stack, for MongoDB, Express, Angular, and Node.js), but millions of developers are still writing Java and .Net, and Ruby and Python have significant followings. Linux is arguably the leading platform, but commercial Unix and Microsoft platforms are still strong. And, periodically it's reported that the majority of the world's transactions **still** run on COBOL-based systems.

However, in the past few years, some powerful infrastructure concepts have solidified that are independent of particular platforms:

- "Cloud"-based technology services
- Automation and "infrastructure as code"
- The centrality of source control
- The importance of package management
- Policy-based infrastructure management

(We'll get to test-driven development, pipeline automation & DevOps in the next chapter.)

This might seem like a detour - you are in a hurry to start writing code! But industry practice is clear. You check your code into source control from Day One. You define your server configurations as recipes, manifests, or at least shell scripts, and check those definitions into source control as well. You keep track of what you have downloaded from the Internet and what version of stuff you are using, through package management (which uses different tools than source control). Always downloading

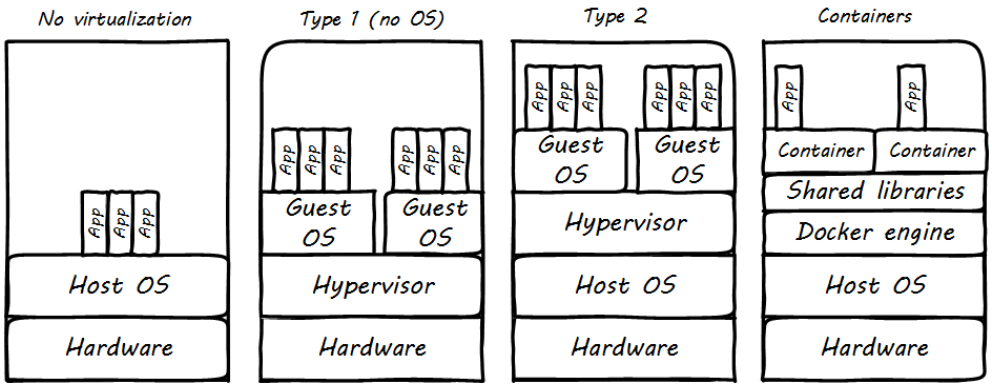


Figure 2.8: Virtualization types

Note
Virtualization was predicted in the earliest theories that led to the development of computers. Turing and Church realized that any general purpose computer could emulate any other. Virtual systems have existed in some form since at latest 1967 - only 20 years after the first fully functional computers. And yes, you can run computers within computers within computers with virtualization. They get slower and slower the more levels you go in, but the logic still works.

Why is virtualization important?

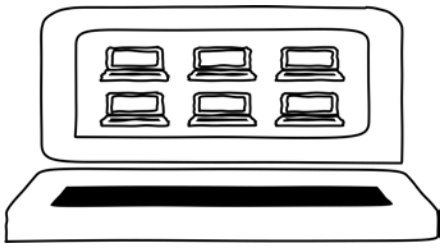


Figure 2.7: Virtualization is computers within a computer

Virtualization!capacity benefits of

Virtualization attracted business attention as a means to consolidate computing workloads. For years, companies would purchase servers to run applications of various sizes, and in many cases the computers were badly underutilized. Because of configuration issues (legitimate) and an overabundance of caution (questionable), average utilization in a pre-virtualization data center might average 10-20%. That's up to 90% of the computer's capacity being wasted (see Figure 2.9).

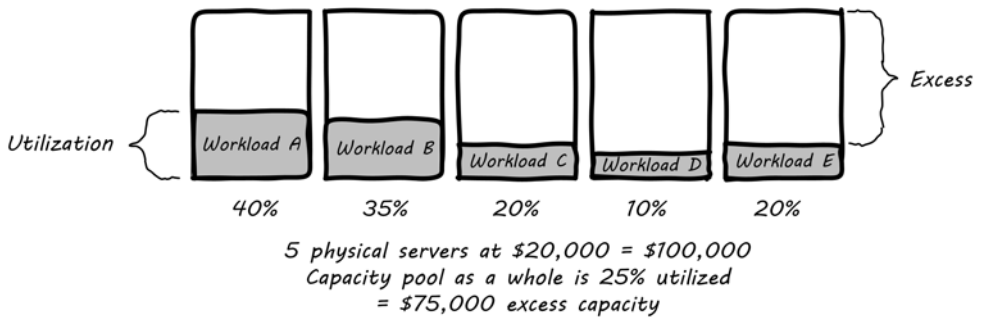


Figure 2.9: Inefficient utilization

The above figure is a simplification. Computing and storage infrastructure supporting each application stack in the business were sized to support each workload. For example, a payroll server might run on a different infrastructure configuration than a data warehouse server. Large enterprises needed to support hundreds of different infrastructure configurations, increasing maintenance and support costs.

The adoption of virtualization allowed businesses to compress multiple application workloads onto a smaller number of physical servers (see Figure 2.10).

Note
For illustration only. A utilization of 62.5% might actually be a bit too high for comfort, depending on the variability and criticality of the workloads.

In most virtualized architectures, the physical servers supporting workloads share a consistent configuration, which made it easy to add and remove resources from the environment. The virtual machines may still vary greatly in configuration, but the fact of virtualization makes managing that easier - the virtual machines can be easily copied and moved, and increasingly can be defined as a form of code (see next section).

Virtualization thus introduced a new design pattern into the enterprise where computing and storage infrastructure became commoditized building blocks supporting an ever-increasing array of services. But what about where the application is large and virtualization is mostly overhead? Virtualization still may make sense in terms of management consistency and ease of system recovery.

Virtualization, managed services, and cloud

Companies have always sought alternatives to owning their own computers. There is a long tradition of managed services, where applications are built out by a customer and then their management is outsourced to a third party. Using fractions of mainframe “time-sharing” systems is a practice that dates back decades. However, such relationships took effort to set up and manage, and might even require bringing

physical tapes to the third party (sometimes called a "service bureau.") Fixed price commitments were usually high (the customer had to guarantee to spend X dollars.) Such relationships left much to be desired in terms of responsiveness to change.

As computers became cheaper, companies increasingly acquired their own data centers, investing large amounts of capital in high-technology spaces with extensive power and cooling infrastructure. This was the trend through the late 1980s to about 2010, when Cloud computing started to provide a realistic alternative with true "pay as you go" pricing, analogous to electric metering.

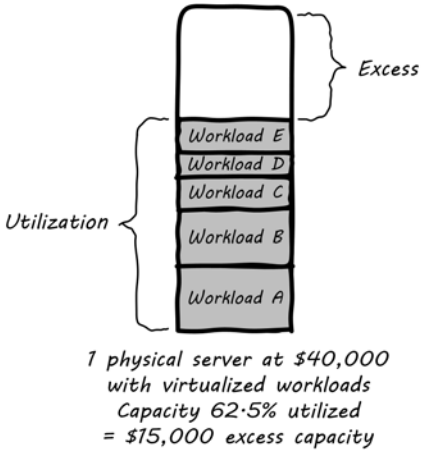


Figure 2.10: Efficiency through virtualization

The idea of running IT completely as a utility service goes back at least to 1965 and the publication of *The Challenge of the Computer Utility*, by Douglas Parkhill (see Figure 2.11). While the conceptual idea of Cloud and utility computing was foreseeable fifty years ago, it took many years of hard-won IT evolution to support the vision. Reliable hardware of exponentially increasing performance, robust open-source software, Internet backbones of massive speed and capacity, and many other factors converged towards this end.

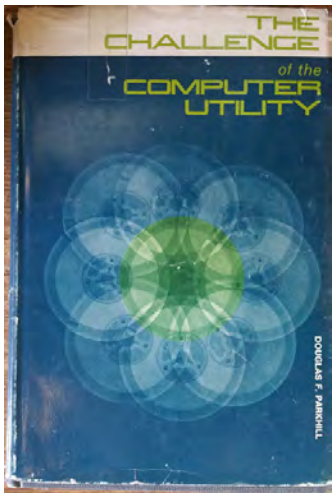


Figure 2.11: Initial statement of Cloud computing

However, people store data - often private - on computers. In order to deliver compute as a utility, it is essential to segregate each customer's workload from all others. This is called *multi-tenancy*. In multi-tenancy, multiple customers share physical resources that provide the illusion of being dedicated.

Note

The phone system has been multi-tenant ever since they got rid of party lines. A party line was a shared line where anyone on it could hear every other person.

In order to run compute as a utility, multi-tenancy was essential. This is different from electricity (but similar to the phone system). As noted elsewhere, one watt of electric power is like any other and there is less concern for leakage or unexpected interactions. People's bank balances are not encoded

Test-driven development enables the next major practice, that of refactoring. Refactoring is how you address technical debt. What is technical debt? Technical debt is a term coined by Ward Cunningham, and is now defined by Wikipedia as

. . . the eventual consequences of poor system design, software architecture or software development within a codebase. The debt can be thought of as work that needs to be done before a particular job can be considered complete or proper. If the debt is not repaid, then it will keep on accumulating interest, making it hard to implement changes later on. . . Analogous to monetary debt, technical debt is not necessarily a bad thing, and sometime technical debt is required to move projects forward. [272]

Test driven development ensures that the system's functionality remains consistent, while refactoring provides a means to address technical debt as part of ongoing development activities. Prioritizing the relative investment of repaying technical debt versus developing new functionality will be examined in future sections, but at least you now know the tools and concepts.

We discuss technical debt further in Chapter 12.

Continuous integration

Version control, again: branching and merging

> Oddly enough, it seems that when you run into a painful activity, a good tip is to do it more often.
>
> — Martin Fowler *Foreword to Paul Duvall's Continuous Integration*

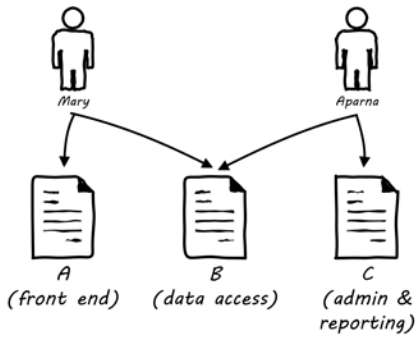


Figure 3.8: Two developers, one file

As systems engineering approaches transform to Cloud and Infrastructure as Code, a large and increasing percentage of IT work takes the form of altering text files and tracking their versions. We have seen this in the previous chapter, with artifacts such as scripts being created to drive the provisioning and configuring of computing resources. Approaches which encourage ongoing development and evolution are increasingly recognized as less risky, since systems do not respond well to big "batches" of change. An important concept is that of "continuous integration," popularized by Paul Duvall in his book of the same name [81].

In order to understand why continuous integration is important, it is necessary to further discuss the concept of source control and how it is employed in real world

settings. Imagine Mary have been working for some time with her partner Aparna in their startup (or on a small team) and they have three code modules (see Figure 3.8). Mary is writing the web front end (file A), Aparna is writing the administrative tools and reporting (file C), and they both partner on the data access layer (file B). The conflict of course arises on the file B that they both need to work on. A and C are mostly independent of each other, but changes to any part of B can have an impact on both their modules.

If changes are frequently needed to B, and yet they cannot split it into logically separate modules, they have a problem; they cannot both work on the same file at the same time. They are each concerned that the other does not introduce changes into B that "break" the code in their own modules A and C.

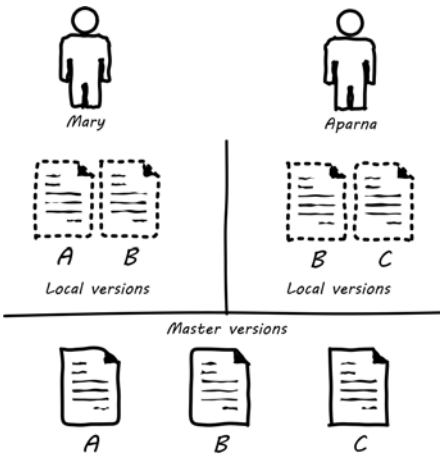


Figure 3.9: File B being worked on by 2 people

In smaller environments, or under older practices, perhaps there is no conflict, or perhaps they can agree to take turns. But even if they are taking turns, Mary still needs to test her code in A to make sure it's not been broken by changes Aparna made in B. And what if they really both need to work on B (see Figure 3.9) at the same time?

Now, because they took this book's advice and didn't start developing until they had version control in place, each of them works on a "local" copy of the file (see illustration "File B being worked on by 2 people").

That way, they can move ahead on their local workstations. But when the time comes to combine both of your work, they may find themselves in "merge hell." They may have chosen very different approaches to solving the same problem, and code may need massive revision to settle on one code base. For example, in the accompanying illustration, Mary's changes to B are represented by triangles and Aparna's are represented by circles. They each had a local version on their workstation for far too long, without talking to each other.

Breaking a system apart by "layer" (e.g. front end versus data access) does not scale well. Microservices approaches encourage keeping data access and business logic together in functionally cohesive units. More on this in future chapters. But in this example, both developers are on the same small team. It is not always possible (or worth it) to divide work to keep two people from ever needing to change the same thing.

In the diagrams, we represent the changes graphically; of course, with real code, the different graphics represent different development approaches each person took. For

example, Mary had certain needs for how errors were handled, while Aparna had different needs.

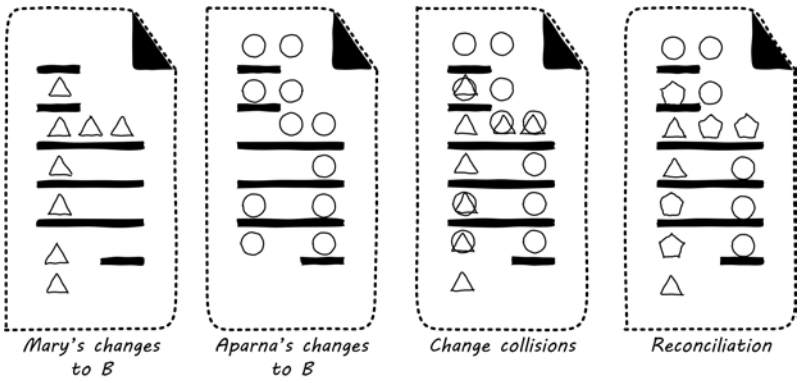


Figure 3.10: Merge hell

In Figure 3.10, where triangles and circles overlap, Mary and Aparna painstakingly have to go through and put in a consolidated error handling approach, so that the code supports both of their needs. The problem of course is now there are three ways errors are being handled in the code. This is not good, but they did not have time to go back and fix all the cases. This is a classic example of technical debt.

Suppose instead that they had been checking in every day. They can identify the first collision quickly (see Figure 3.11), and have a conversation about what the best error handling approach is. This saves them **both** the rework of fixing the collisions, **and** the technical debt they might have otherwise accepted:

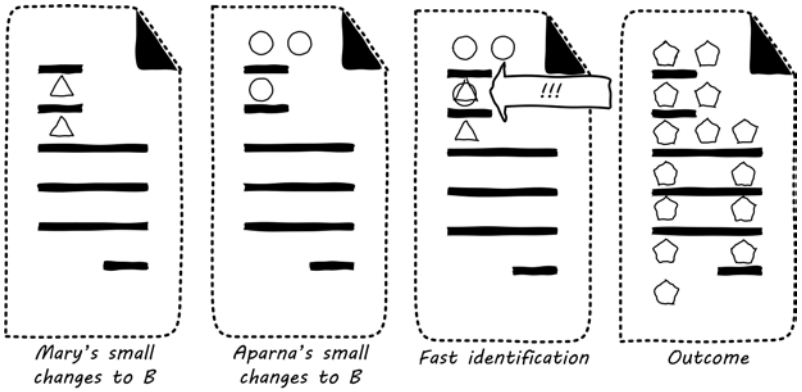


Figure 3.11: Catching errors quickly is valuable

These problems have driven the evolution of for decades. In previous methods, to develop a new release, the code would be copied into a very long-lived "branch" (a version of the code to receive independent enhancement). Ongoing "maintenance"

fixes of the existing code base would also continue, and the two code bases would inevitably diverge. Switching over to the "new" code base might mean that once-fixed bugs (bugs that had been addressed by maintenance activities) would show up again, and of course this would not be acceptable. So, when the newer development was complete, it would need to be merged back into the older line of code, and this was rarely if ever easy (again, "merge hell"). In a worst case scenario, the new development might have to be redone.

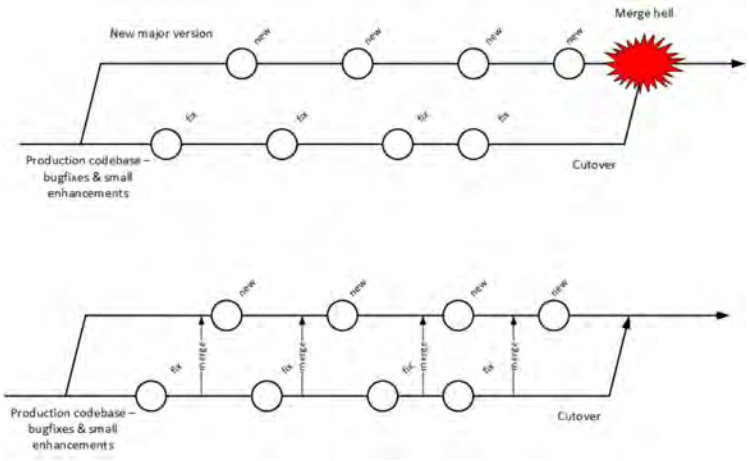


Figure 3.12: Big bang vs. continuous integration

Enter continuous integration (see Figure 3.12). As presented in [81] the key practices (you will notice similarities to the pipeline discussion) include:

- Developers run private builds including their automated tests before committing to source control
- Developers check in to source control at least daily (hopefully we have been harping on this enough that you are taking it seriously by now).
  - Distributed version control systems such as git are especially popular, although older centralized products are starting to adopt some of their functionality
  - Integration builds happen several times a day or more on a separate, dedicated machine
- 100% of tests must pass for each build. Fixing failed builds is the highest priority.
- A package or similar executable artifact is produced for functional testing
- A defined package repository exists as a definitive location for the build output.

These practices are well developed and represent a highly evolved understanding gained through the painful trial and error of many development teams over many years. Rather than locking C so that only one person can work on it at a time, it's

been found that the best approach is to allow developers to actually make multiple copies of such a file or file set and work on them simultaneously. Wait, you say. How can that work?

This is the principle of continuous integration at work. If the developers are continually pulling each other's work into their own working copies, and continually testing that nothing has broken, then distributed development can take place. So, if you are a developer, the day's work might be as follows:

8 AM: check out files from master source repository to a local branch on your workstation. Because files are not committed unless they pass all tests, you know that you are checking out clean code. You pull user story (requirement) that you will now develop.

8:30 AM: You define a test and start developing the code to fulfill it.

10 AM: You are closing in on wrapping up the first requirement. You check the source repository. Your partner has checked in some new code, so you pull it down to your local repository. You run all the automated tests and nothing breaks, so you're fine.

10:30: You complete your first update of the day; it passes all tests on your workstation. You commit it to the master repository. The master repository is continually monitored by the build server, which takes the code you created and deploys it, along with all necessary configurations, to a dedicated build server (which might be just a virtual machine or transient container). All tests pass there (the test you defined as indicating success for the module, as well as a host of older tests that are routinely run whenever the code is updated.

11:00: Your partner pulls your changes into their working directory. Unfortunately, some changes you made conflict with some work they are doing. You briefly consult and figure out a mutually acceptable approach.

Controlling simultaneous changes to a common file is only one benefit of continuous integration. When software is developed by teams, even if each team has its own artifacts, the system often fails to "come together" for higher-order testing to confirm that all the parts are working correctly together. Discrepancies are often found in the interfaces between components; when component A calls component B, it may receive output it did not expect and processing halts. Continuous integration ensures that such issues are caught early.

Build choreography

Go back to the pipeline picture and consider step 4. While we discussed version control, package management, and deployment management in Chapter 2, this is our first encounter with build choreography.

DevOps and continuous delivery call for automating everything that can be automated. This goal led to the creation of build choreography managers such as Hudson, Jenkins, Travis CI, and Bamboo. Build managers may control any or all of the following steps:

- Detecting changes in version control repositories and building software in response

- Alternately, building software on a fixed (e.g. nightly) schedule
- Compiling source code and linking it to libraries
- Executing automated tests
- Combining compiled artifacts with other resources into installable packages
- Registering new and updated packages in the package management repository, for deployment into downstream environments.
- In some cases, driving deployment into downstream environments, including production. (This can be done directly by the build manager, or through the build manager sending a message to a deployment management tool.)

Build managers play a critical, central role in the modern, automated pipeline and will likely be a center of attention for the new digital professional in their career.

Releasing software

Continuous deployment

(see Figure 3.13)

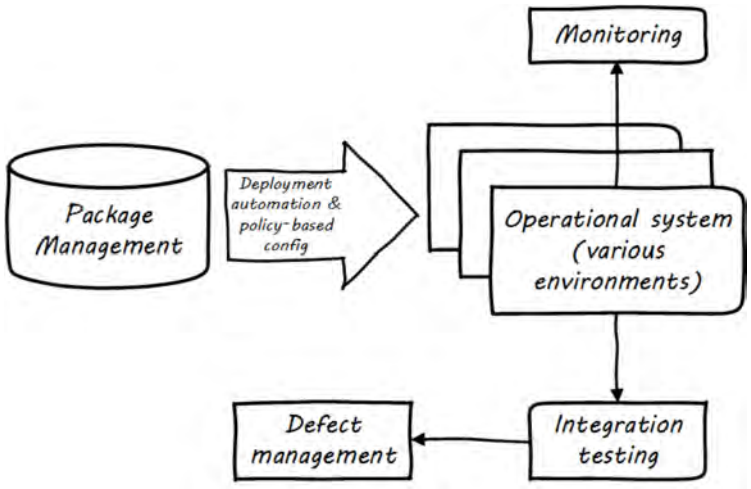


Figure 3.13: Deployment

Once software is compiled and built, the executable files that can be installed and run operationally should be checked into a Package Manager. At that point, one can take the last mile step and deploy the now tested and built software to pre-production or production environments (see Figure 3.13). The software can undergo usability testing, load testing, integration testing, and so forth. Once those tests are passed, it can be deployed to production.

Part II

Team

(CLD), it is saying that Change and Stability are opposed - the more we have of one, the less we have of the other. This is true, as far as it goes - most systems issues occur as a consequence of change; systems that are not changed in general do not crash as much.

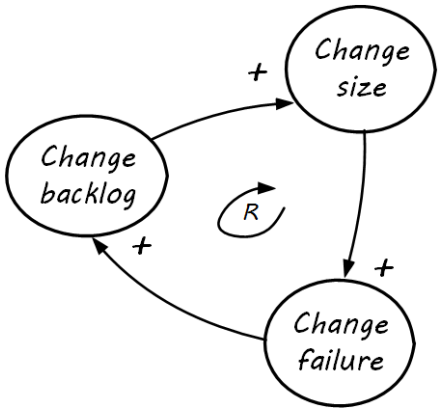


Figure 3.23: Change vicious cycle

The trouble with viewing change and stability as diametrically opposed is that change is inevitable. If simple delaying tactics are put in, these can have a negative impact on stability, as in Figure 3.23. What is this diagram telling us? If the owner of the system tries to prevent change, a larger and larger backlog will accumulate. This usually results in larger and larger scale attempts to clear the backlog (e.g. large releases or major version updates.) These are more risky, and increase the likelihood of change failure. When changes fail, the backlog is not cleared and continues to increase, leading to further temptation for even larger changes.

How do we solve this? Decades of thought and experimentation have resulted in continuous delivery and DevOps, which can be shown in terms of system thinking in Figure 3.24.

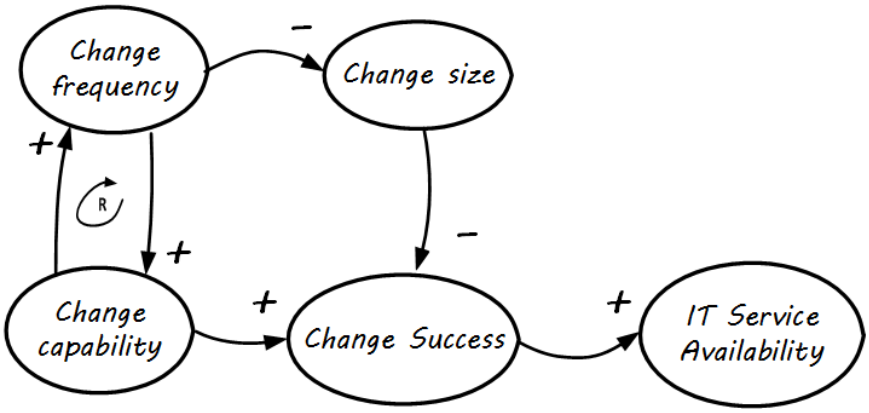


Figure 3.24: The DevOps consensus

To summarize a complex set of relationships:

- As change occurs more frequently, it enables smaller change sizes.
- Smaller change sizes are more likely to succeed (as change size goes up, change success likelihood goes down, hence it is a *balancing* relationship).

Chapter 4

Product Management

Introduction

Product Management?

"Product management?" In a book on IT management? Those of you with industry experience, especially backgrounds in project-based enterprise software development, may be unfamiliar with the term. However, a focus on product development is one of the distinguishing features of Agile development, even if that development is taking place in a larger enterprise context.

As you grow your company, you are bringing more people in. You become concerned that they need to share the same vision that inspired you to create this company. This is the goal of product management as a formalized practice.

Product strategy was largely tacit in Part I. As the founder, you used product management and discovery practices, and may well be familiar with the ideas here, but the assumption is that you did not explicitly **formalize** your approach to them. Now you need a more prescriptive and consistent approach to discovering, defining, designing, communicating, and executing a product vision across a diverse team.

In this chapter, we will define and discuss product management, and distinguish it from project and process management. We will cover how product teams are formed and what practices and attitudes you should establish quickly.

We will discuss a number of specific schools of thought and practices, including Gothelf's Lean UX, Scrum, and more specific techniques for product "discovery." Finally, we will discuss the concepts of design and design thinking.

Chapter 4 outline

- Why product management?
  - The product vision

IT systems started by serving narrow purposes, often "back office" functions such as accounting or materials planning. Mostly, such systems were managed as projects assembled on a temporary basis, resulting in the creation of a system to be "thrown over the wall" to operations. Product management, on the other hand, is concerned with the entire lifecycle. The product manager (or owner, in Scrum terms) cares about the vision, its execution, the market reaction to the vision (even if an internal market), the health, care and feeding of the product, and the product's eventual sunset or replacement.

In the enterprise IT world, "third party" vendors (e.g. IBM) providing the back office systems had product management approaches, but these were **external** to the IT operations. Nor were IT-based product companies as numerous forty years ago as they are today; as noted in chapter 1, with digital transformation, the digital component of modern products continues to increase to the point where it's often not clear whether a product is "IT" or not.

Figure 4.1: Product design session

Reacting to market feedback and adapting product direction is an essential role of the product owner. In the older model, feedback was often unwelcome, as the project manager typically was committed to the open-loop dead reckoning of the project plan and changing scope or direction was seen as a failure, more often than not.

Now, it's accepted that systems evolve, perhaps in unexpected directions. Rapidly testing, failing fast, learning, and pivoting direction are all part of the lexicon, at least for market-facing IT-based products. And even back-office IT systems with better understood scope are being managed more as systems (or products) with lifecycles, as opposed to transient projects. (See the Amazon discussion, below.)

So, what is product management and what does it mean for your team? [26]

Defining Product Management

In order to define product management, we first need to define product. In Chapter 1, we established that products are goods, services, or some combination, with some feature that provides value for some consumer. BusinessDictionary.com defines it thus:

> [A Product is] A good, idea, method, information, object or service created as a result of a process and serves a need or satisfies a want. It has a

In Chapter 3, we needed to consider the means for describing system intent. Even as a bare-bones startup, some formalization of this starts to emerge, at the very least in the form of test-driven development (see Figure 4.9).

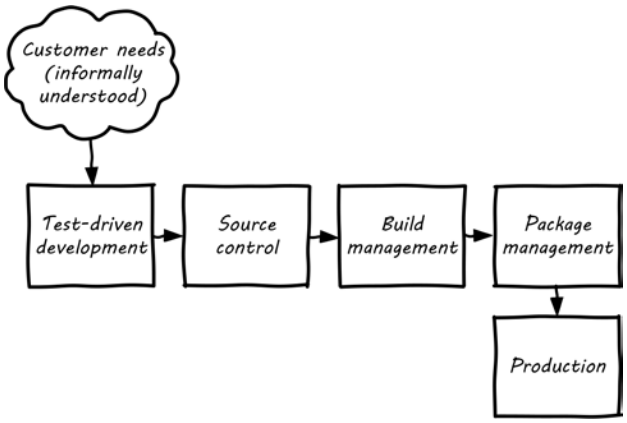


Figure 4.9: Product discovery tacit

But, the assumption in our emergence model is that more formalized product management emerges with the formation of a team. As a team, we now need to expand "upstream" of the core delivery pipeline, so that we can collaborate and discover more effectively. Notice the grey box in Figure 4.10.

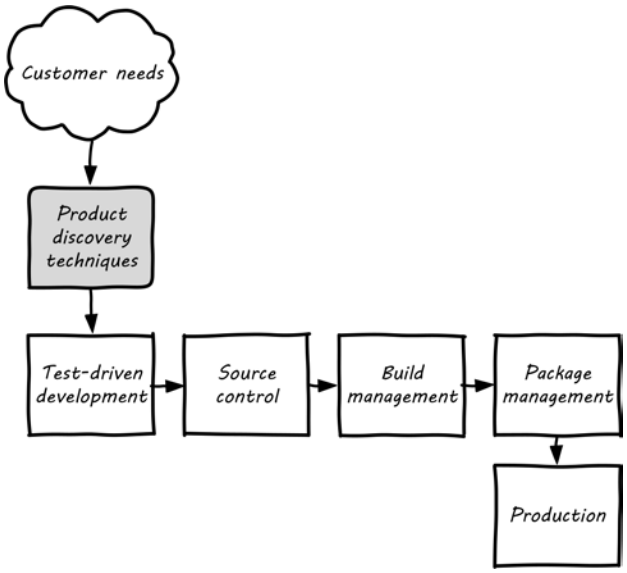


Figure 4.10: Product discovery explicit

The most important foundation for your newly formalized product discovery capability is that it must be **empirical and hypothesis-driven**. Too often, product strategy is based on the HIPPO: The Highest Paid Person's Opinion (Figure 4.11 [30]).

The problem with relying on "gut feel" or personal opinions is that people — regardless of experience or seniority — perform poorly in assessing the likely outcome of their product ideas. Some well known research on this topic was conducted by Microsoft's Ronny Kohavi. In this research, Kohavi and team determined that "only about 1/3 of ideas improve the metrics they were designed to improve." [152] As background, the same report cites that:

Figure 4.11: Beware of HIPPO-based product discovery

- "Netflix considers 90% of what they try to be wrong"
- "75 percent of important business decisions and business improvement ideas either have no impact on performance or actually hurt performance" according to Qualpro (a consultancy specializing in controlled experiments)

It is therefore critical to establish a strong practice of data-driven experimentation when forming a product team, and avoid any cultural acceptance of "gut feel" or deferring to HIPPOs. This can be a difficult transition for the company founder, who has until now served as the *de facto* product manager.

A useful framework, similar to Lean Startup is proposed by Spotify, in the "DIBB" model:

- Data
- Insight
- Belief
- Bet

Data leads to insight, which leads to a hypothesis that can be tested (i.e., "bet" on - testing hypotheses is not free). We discuss issues of prioritization further in Chapter 5, in the section on cost of delay.

Don Reinertsen (who we will read more about in the next chapter) emphasizes that such experimentation is inherently *variable*. We can't develop experiments with any sort of expectation that they will always succeed. We might run 50 experiments, and only have 2 succeed. But if the cost of each experiment is $10,000, and the two that succeeded earned us $1 million each, we gained:

```
$ 2,000,000
$ - 480,000
-----------
$ 1,520,000
```

or all of the organization, so what would be the point? A shared white board in a public location might be all that is needed (see Figure 5.3). This gives the team a "shared mental model" of who is doing what.

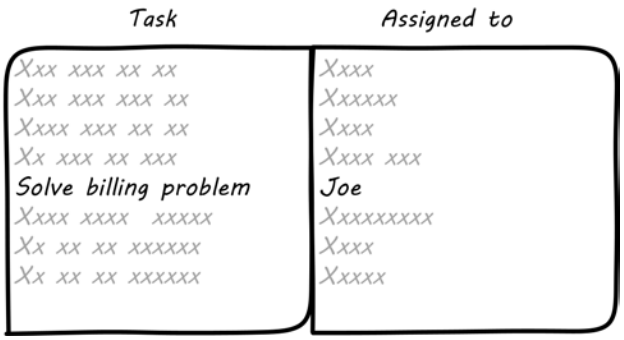


Figure 5.3: Common list

The design of the task board above has some issues, however. After the team gets tired of erasing and rewriting the tasks and their current assignments, they might adopt something more like this:

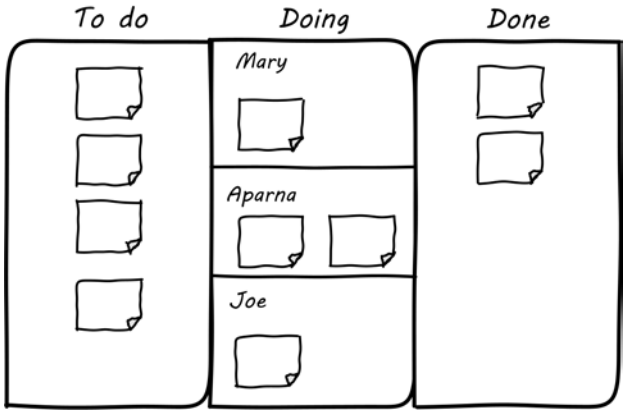


Figure 5.4: Simple task board

The board itself might be a white board, or a cork bulletin board with push pins (see Figure 5.4). The notes could be sticky, or index cards. There are automated solutions as well. The tool doesn't really matter. The important thing is that, at a glance, the entire team can see its flow of work and who is doing what.

This is sometimes called a "Kanban board," although David Anderson (originator of the Kanban software method [12]) himself terms the basic technique a "card wall." It also has been called a "Scrum Board." The board at its most basic is not specific to either methodology. The term "Kanban" itself derives from Lean manufacturing principles; we will cover this in depth in the next section. The basic board is widely

used because it is a powerful artifact. Behind its deceptive simplicity is considerable industrial experience and relevant theory from operations management and human factors. However, it has scalability limitations. What if the team is not all in the same room? We will cover this and related issues in Part III.

Important

The card wall or Kanban board is the first channel we have for *demand management.* Demand management is a term meaning "understanding and planning for required or anticipated services or work." Managing day to day incoming work is a form of demand management. Capturing and assessing ideas for next year's project portfolio (if you use projects) is also demand management at a larger scale.

Learning from manufacturing

Instructor's note

The concepts of queuing and work in process are critical to the rest of this book. Recommend classroom exercises and additional reading to ensure that they are well understood by students. The *Phoenix Project* and *The Goal* are excellent, entertaining books that use novelization to illustrate these principles.

Manufacturing? What do digital professionals have to learn from that? "Our work is not an assembly line!," is one frequently heard response. It is true that we need to be careful in drawing the **right** lessons from manufacturing, but there is a growing consensus on how to do this. Please keep an open mind as you read this chapter.

Kanban and its Lean origins

To understand Kanban let's discuss Lean briefly. We've had passing mention of Lean already in this book. But what is it?

Important

Lean is important. Regardless of your intended career path, it is advisable to read the great Lean classics, including The Machine That Changed the World, Lean Thinking, The Toyota Way, and Ohno's own Toyota Production System. Toyota Kata is a more recent, in-depth analysis of Toyota's culture.

Lean is a term invented by American researchers who investigated Japanese industrial practices and their success in the 20th century. After the end of World War II, no-one expected the Japanese economy to recover the way it did. The recovery is credited to practices invented by Taiichi Ohno (see Figure 5.5 [37]) and Shigeo Shingo at Toyota. These practices included:

- Respect for people
- Limiting work in process
- Small batch sizes (driving towards “single piece flow”)
- Just-in-time production
- Decreased cycle time

Credit for Lean is also given to U.S. thinkers such as W.E. Deming, Peter Juran, and the theorists behind the Training Within Industry methodology, who all played influential roles in shaping the industrial practices of post-war Japan.

Figure 5.5: Lean pioneer Taichi Ohno

Kanban is a term originating from Lean and the Toyota Production System. Originally, it signified a “pull” technique in which materials would only be transferred to a given workstation on a definite signal that the workstation required the materials. This was in contrast to “push” approaches where work was allowed to accumulate on the shop floor, on the (now discredited) idea that it was more “efficient” to operate workstations at maximum capacity.

Factories operating on a “push” model found themselves with massive amounts of inventory (work in process) in their facilities. This tied up operating capital and resulted in long delays to shipment. Japanese companies did not have the luxury of large amounts of operating capital, so they started experimenting with ”single-piece flow.” This led to a number of related innovations, such as the ability to re-configure manufacturing machinery much more quickly than U.S. factories were capable of.

David J. Anderson was a product manager at Microsoft who was seeking a more effective approach to managing software development. In consultation with Don Reinertsen (introduced in the below) he applied the original concept of Kanban to his software development activities [12].

Scrum (covered in the previous chapter) is based on a rhythm with its scheduled sprints, for example every two weeks. In contrast, Kanban is a continuous process with no specified rhythm (also known as *cadence*). Work is “pulled” from backlog into active attention as resources are freed from previous work. This is perhaps the most important aspect of Kanban - the idea that **work is not accepted until there is capacity to perform it.**

You may have a white board covered with sticky notes, but if they are stacked on top of each other with no concern for worker availability, you are not doing Kanban. You are accepting too much work in process and you are likely to encounter a “high-queue state” in which work becomes slower and slower to get done. (More on queues below.)

The Theory of Constraints

Eliyahu Moshe Goldratt was an Israeli physicist and management consultant, best known for his pioneering work in management theory, including *The Goal*, which is a best-selling business novel frequently assigned in MBA programs. It and Goldratt's other novels have had a tremendous effect on industrial theory, and now, digital management. One of the best known stories in *The Goal* centers around a Boy Scout march. Alex, the protagonist struggling to save his manufacturing plant, takes a troop of Scouts on a ten mile hike. The troop has hikers of various speeds, yet the goal is to arrive simultaneously. As Alex tries to keep the Scouts together, he discovers that the slowest, most overweight scout (Herbie) also has packed an unusually heavy backpack. The contents of Herbie's pack are redistributed, speeding up both Herbie and the troop.

Author's note: Gene Kim and The Phoenix Project

Between 2005 and 2012, I was a lead enterprise architect at Wells Fargo Bank, primarily concerned with IT delivery capabilities such as portfolio and service management. One day around 2007, I arrived at my office to find an envelope from my friend Gene Kim, then CTO of Tripwire. Gene and I had been corresponding for some years on high-performing IT, IT process improvement, and related topics. In the envelope was a copy of a book called *The Goal*, by Eli Goldratt [103]. I was a little mystified, but after reading the book I began to understand.

Figure 5.6: Gene Kim

Gene saw the potential of the Theory of Constraints in understanding certain aspects of information technology management, and used it as a template to write another remarkable and influential book, *The Phoenix Project* [145]. Rather than a manufacturing plant, the Phoenix Project centers on the struggles of the IT team at a medium-sized automotive parts manufacturer and retailer. From a state of chaos, uncontrolled work in process and resource constraints, the team applies Lean, Agile, and DevOps techniques to great effect. In my view, *The Phoenix Project* is one of the most important works in the history of IT and digital management, and is also an enjoyable novel. I am honored to have been one of the original reviewers. If you are considering a career in IT or digital, it is essential reading. See especially Chapter 30 for an interesting discussion of manufacturing lessons in an IT context.

This story summarizes the Goldratt approach: finding the "constraint" to production (his work as a whole is called the Theory of Constraints). In Goldratt's view, a system is only as productive as its constraint. At Alex's factory, it's found that the "constraint" to the overall productivity issues is the newest computer-controlled machine tool — one that could (in theory) do the work of several older models but was now jeopardizing the entire plant's survival. This novelization parallels in important regards actual Lean case studies on the often-negative impact of such capital-intensive

Part III

Team of Teams

Figure 6.18: All hands meeting at NASA Goddard

Team of teams

Team of Teams: New Rules of Engagement for a Complex World is the name of a 2015 book by General Stanley McChrystal, describing his experiences as the commander of Joint Special Operations Command in the Iraq conflict. It describes how the U.S. military was being beaten by a foe with inferior resources, and its need to shift from a focus on mechanical efficiency to more adaptable approaches. The title is appropriate for this section, as moving from "team" to "team of teams" is one of the most challenging transitions any organization can make.

Scenario

You are now a "team of teams," at a size where face to face communication is increasingly supplemented by other forms of communication and coordination. Your teams are all good, and get results, but in different ways. You need some level of coordination and not everyone can talk to everyone; people are no longer co-located and there may be different schedules involved.[55]

You now have multiple products. As you scale up, you now must split your products into features and components (the y-axis of the AKF scaling cube). Then as you move from your first product to adding more, even more organizational evolution is required. You try to keep your products from developing unmanageable interdependencies, but this is an ongoing challenge. Tensions between various teams are starting to emerge. You are seeing more and more specialization in your organization. You see a tendency of specialists to identify more with their field than with the needs of your customers and your business. There is an increasing desire among your stakeholders and executives for control and predictability. Resources are limited and always in contention. You are considering various frameworks for managing your organization. As we scale, however, we need to remember that our highest value is found in

fast-moving, committed, multi-skilled teams. Losing sight of that value is a common problem for growing organizations. This is where it gets hard.

As you become a manager of managers, your concerns again shift. In Part II, you had to delegate product management (are they building the right thing?) and take concern for basic work management and digital operations. Now, as your organization grows, you are primarily a manager of managers, concerned with providing the conditions for your people to excel:

- Defining how work is executed, in terms of decision rights, priorities, and conflicts
- Setting the organizational mission and goals that provide the framework for making investments in products and projects
- Instituting labor, financial, supply chain, and customer management processes and systems
- Providing facilities and equipment to support digital delivery
- Resolving issues and decisions escalated from lower levels in the organization

(influenced by [202].)

New employees are bringing in their perspectives, and the more experienced ones seem to assume that the company will use "projects" and "processes" to get work done. There are no shortage of contractors and consultants all advocating various flavors of process and project management, some advocating older approaches and "frameworks" and others proposing newer Agile & Lean perspectives. However, the ideas of process and project management are occasionally called into question by both your employees and various "thought leaders," and it's all very confusing.

Welcome to the coordination problem. We need to understand where these ideas came from, how they relate to each other, and how they are evolving in a digitally transforming world.

Here is an overview of Part III's structure:

Special section: Scaling the organization and its work

Digital professionals use a number of approaches to defining and managing work at various scales. Our initial progression from product, to work, to operations management can be seen as one dimension. We consider a couple of other dimensions as a basis for ordering Part III.

Chapter 7: Coordination

Going from one to multiple teams is hard. No matter how you structure things, there are dependencies requiring coordination. How do you ensure that broader goals are met when teams must act jointly? Some suggest project management, while others argue that you don't need it any more - it's all about continuous flow through loosely-coupled product organizations. But you've seen that your most ambitious ideas require some kind of choreography, and that products and projects need certain resources and services delivered predictably. When is work repeatable? When is it

unique? Understanding the difference is essential to your organization's success. Is variability in the work always bad? These are questions that have preoccupied management thinkers for a long time.

Chapter 8: Planning and investment

Each team also represents an investment decision. You now have a portfolio of features, and/or products. You need a strategy for choosing among your options and planning — at least at a high level — in terms of costs and benefits. Some of you may be using project management to help manage your investments. Your vendor relationships continue to expand; they are another form of strategic investment, and you need to deepen your understanding of matters like Cloud contracts and software licensing. Finally, what is your approach to finance and budgeting?

Note

In terms of classic project methodology, chapter 8 includes project initiating and planning. Execution, monitoring, and control of day to day work are covered in Chapter 7. The seemingly backwards order is deliberate, in keeping with the emergence model.

Chapter 9: Organization and culture

You're getting big. In order to keep growing, you have had to divide your organization. How are you formally structured? In terms of your market, or your resources? How are people grouped, and to whom do they report, with what kind of expectations? Finally, what is your approach to bringing new people into your organization? What are the unspoken assumptions that underly your daily work — in other words, what is your culture? Does your culture support high performance, or the opposite? How can you measure and know such a thing?

Important

Part III, like the other parts, needs to be understood as a unified whole. In reality, growing companies struggle with the issues in all three chapters simultaneously.

Special section: Scaling the organization and its work

> Avoid large projects. Start small and quickly develop a product with the minimum functionality… If you have to employ a large project, scale slowly and grow the project organically by adding one team at a time. Starting with too many people causes products to be overly complex, making future product updates time-consuming and expensive.
>
> — Roman Pichler *Agile Product Management with Scrum*

As we begin the second half of this book, consider Pichler's advice above. We have spent chapters 1-6 (the first half of the book) thinking mainly in terms of one product and its dimensions. We are scaling now because we must; we have increasingly diverse product opportunities, or one product that has become so large it must be partitioned in some manner. Or both.

The two dimensions of demand management

To provide a framework for Part III, let's start with this two-dimensional analysis in (Figure 6.19)

You should spend some time reviewing the graphic, which provides a unique way of understanding the work you are now experiencing as a "team of teams" or "manager of managers" in an IT-dependent environment of increasing size and complexity. We've come a long ways since our discussion of work management. By the time we started to formalize operations, we saw that work was tending to differentiate. Still, regardless of the label we put on a given activity, it represents some set of tasks or objectives that real people are going to take time to perform, and expect to be compensated for. It is all demand, requiring management. Remembering this is essential to digital management.

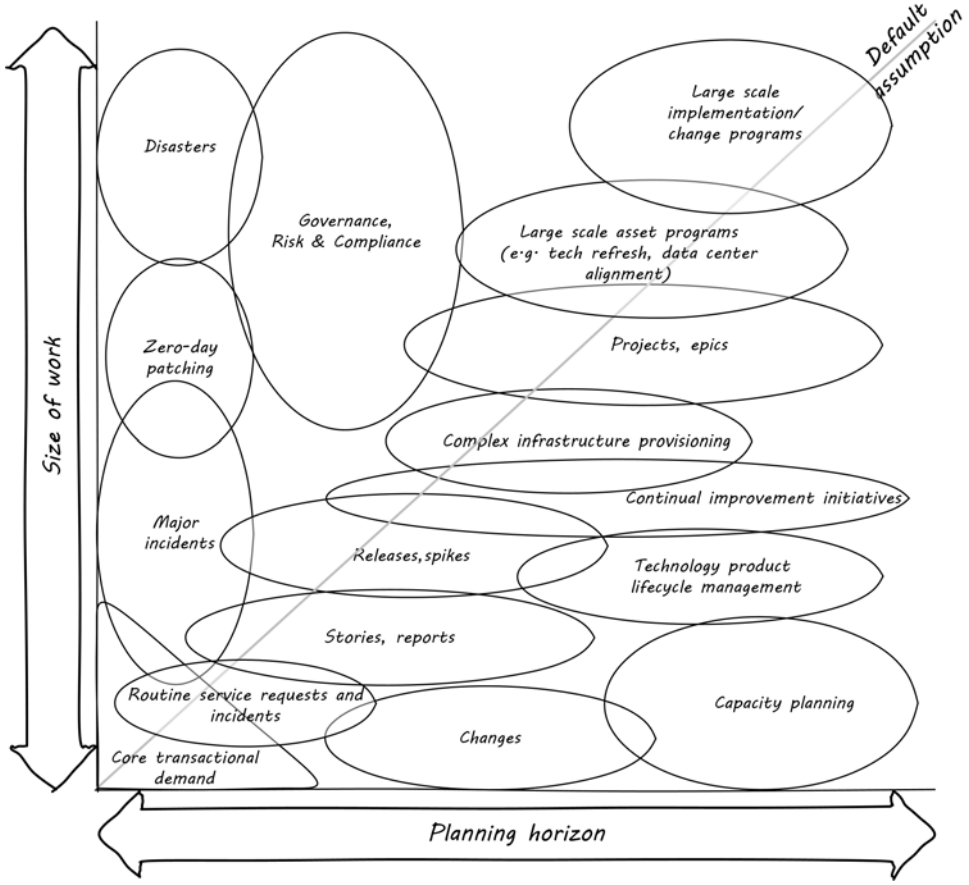


Figure 6.19: Two dimensions of demand management

Let's consider the various forms that demand may take. Understanding these demand forms will also help you develop a deeper understanding of an **architecture of IT management**, a topic I have explored in other works [24]. The diagram has two dimensions:

- Planning
- Granularity

Planning. As an organization scales, there is an increasing span in your time horizon and the scope of work you are considering and executing. From the immediate, "hand-to-mouth" days of your startup, you now must take concern for longer and longer time frames: contracts, regulations, and your company's strategy as it grows all demand this.

Granularity. The terminology you use to describe your work also becomes more diverse, reflecting in some ways the broader time horizons you are concerned with. Requests, changes, incidents, work orders, releases, stories, features, problems, major

incidents, epics, refreshes, products, programs, strategies... there is a continuum of how you think about your organization's work efforts. Mostly, the range of work seems tied to how much planning time you have, but there are exceptions: disasters take a lot of work, but you don't get much advance warning! So size of work is independent of planning horizon.

The bubbles represent a "space" where one is likely to find that kind of work. As indicated by the central diagonal, it reflects an assumption that larger amounts of work are more likely to be planned further in advance. However, this is not always true. A large, unwelcome amount of required work that shows up with no planning is probably a disaster. Desired work (in the form of aggregate transactional demand) may also spike unexpectedly. Transactional demand considered across a long timeframe is capacity management. Table 6.5 lists various examples.

Some forms of work may lead to other forms of work. For example, Projects may manifest as Stories, Releases, and Changes. This complicates the diagram a bit; we don't want to "double-count" work effort. But not all Releases derive from Projects, and not all Project work (especially in complex environments) can be cleanly reduced to a set of smaller tasks.

The final point of this diagram: you only have so much capacity to execute the work it implies. If you have a disaster, or a series of major incidents, this unplanned work may impact your ability to deliver user stories, changes, or even meet transactional demand. Trade-offs must be considered.

Adding a third dimension with Cynefin

This third dimension of variability is challenging to understand and touches on our earlier discussion of systems thinking. A helpful framework to understand it is the Cynefin framework, by Dave Snowden and Cynthia Kurtz [155] (see Figure 6.20 [56]). Cynefin proposes that there are five major domains useful in understanding situations:

- Simple/Obvious
- Complicated
- Complex
- Chaotic
- Disorder

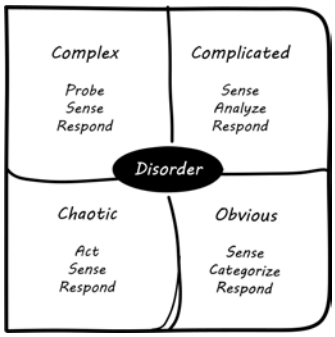


Figure 6.20: Cynefin thinking framework

The **simple or obvious** domain is straightforward, repeatable, and cause and effect are known. The concept of "best practice" applies. The mode of action is to sense, categorize, and respond.

Type of work	Description
Core transactional demand	This is the demand on the fully automated IT system for a given moment of truth: a banking account lookup, a streaming movie, a Human Resources record update
Routine service requests and incidents	Service requests are predefined, process-driven work items, rarely requiring creative thought or analysis. Incidents span a spectrum, but some are simpler and more routine than others, especially those stemming from user misunderstanding or error.
Changes	Changes represent modifications of established IT functionality or state. They represent some definite risk to one or more IT services, which is why they are planned on a longer lead time. However, they ideally remain relatively granular, which helps reduce their risk.
Routine releases, stories, reports	Releases and (in the Agile world) stories represent larger increments of functionality
Projects	A Project is a large, planned amount of work with a defined end date. It might create a Service, which also represents a commitment to a large, ongoing amount of work, perhaps comparable in scope to the Project.
Major incidents	Major incidents by definition are not planned. But they represent a significant amount of work to overcome.

Table 6.5: Work items of varying sizes

The **complicated** domain requires analysis and expertise; there may be several right or at least serviceable answers. Rational thought is possible and cause and effect relationships may be more challenging to understand, but still are applicable. Mode of action is to sense, analyze, and respond.

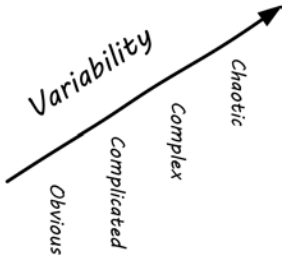


Figure 6.21: Variability as Cynefin domains

The **complex** domain is that of systems thinking. Cause and effect are apparent only in hindsight. Interdependencies complicate action. Reinforcing loops can quickly accelerate, making linear assumptions hazardous, or conversely, counterbalancing loops kick in and prevent desired changes from happening. Mode of action is to probe, sense, and respond ("probe" being to make a small change.) Much of modern product development and DevOps thinking is optimized for this domain, because simple and rational approaches have so frequently failed.

In the **chaotic** domain, cause and effect are not apparent even in hindsight. The situation is completely unpredictable, and action is essential - better to act in any direction than be paralyzed. The mode of action is to act, sense, and respond.

Finally, **disorder** is considered to be the domain you're in when you have not figured out which of the other four applies.

The two dimensional model above does not describe how uncertain work is, however. The predictability of the work is also independent. You might have two projects, both taking the same effort. One of them you were able to predict easily, while the other one was not predictable - more precisely, your expected time, effort and cost was a long way off from what you wound up spending. (Usually in an unfavorable direction.)

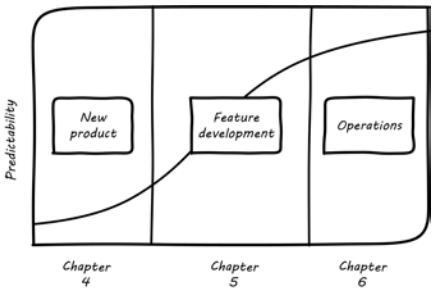


Figure 6.22: Part II: increasing certainty (credit to Cantor)

Part II (Chapters 4-6, which we just finished) can be viewed as a logical progression from the uncertainty of developing a novel product, to the day to day work of building its features, to its predictable operation. The "predictability curve" illustrated in Figure 6.22 [57] increases as the digital product stabilizes and moves to a fully operational state.

This question of predictability, of **the degree to which actuals track estimates** and can be known in advance, will be an ongoing theme throughout Part III. As we scale up, our organization takes on more and more work of all kinds, from highly uncertain to very predictable. Understanding the differences in this "portfolio" of work is essential to managing it correctly. There has always been an element of risk; as a

startup, your success was not guaranteed! You now find that you are managing different classes of risk simultaneously, and "one size fits all" approaches do not work.

You might have a program to upgrade the memory on 80,000 identical Point of Sale terminals across 2,000 retail stores. It's going to take a lot of work; you'll be "rolling trucks" in all 50 states! But you are sure that you can estimate this work with a high degree of accuracy; it has high predictability. In terms of the Cynefin framework (see sidebar), it's an obvious (or maybe complicated) problem. On the other hand, creating a completely new Point of Sale system for your stores is an unpredictable effort. Your original estimate for this large program might be off by orders of magnitude. Its predictability is low. It's a complex problem.

Or perhaps you are writing reports using a well understood database and reporting tool. This work will be likely more predictable work — even if complicated in the Cynefin sense — as compared to developing the first few stories on a completely new architecture. This is true even if the estimated size of the work is the same for both the reports and the new stories. As a dimension, variability is **independent of the size of the work** (although the two may be correlated).

The Betz organizational scaling cube

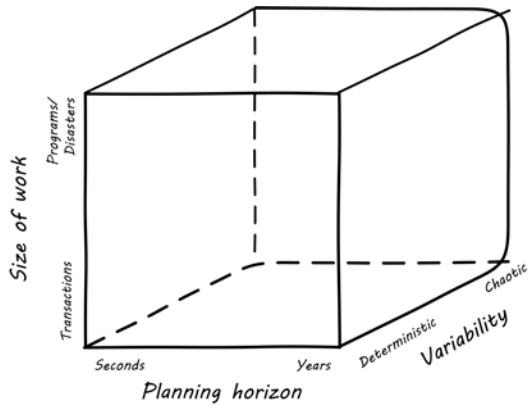


Figure 6.23: Betz organization scaling cube

When we combine the three dimensions:

- Size of work;
- Time horizon; and
- Predictability

we get the Betz organizational scaling cube (see Figure 6.23 [58]). It shows the three dimensions we'll consider throughout Part III. The accompanying cube shows these dimensions visually. The three dimensions represent a space to understand work, resource, and planning as we scale the organization.

The z-axis of variability can be seen as a progression along the first four Cynefin domains (see sidebar). At the origin at lower left, we have predictable, small-grained work occurring in short "planning" horizons (e.g. automated transactions running on computers.) As we scale out to larger domains of work, longer time frames, and greater variability in planning, we encounter the problems of growth, coordination, strategy, and the fundamental uncertainties of operating in a chaotic, competitive world.

Chapter 7

Coordination

Introduction

> coordination (n.) co-ordination, c. 1600, "orderly combination," from French coordination (14c.) or directly from Late Latin coordinationem (nominative coordinatio), noun of action from past participle stem of Latin coordinare "to set in order, arrange," from com- "together" (see com-) + ordinatio "arrangement," from ordo "row, rank, series, arrangement" (see order (n.)). Meaning "action of setting in order" is from 1640s; that of "harmonious adjustment or action," especially of muscles and bodily movements, is from 1855.
>
> — Online Etymology Dictionary

> Agile software development methods were particularly designed to deal with change and uncertainty, yet they de-emphasize traditional coordination mechanisms such as forward planning, extensive documentation, specific coordination roles, contracts, and strict adherence to a pre-specified process [248].
>
> — Diane E. Strode et al *Coordination in co-located agile software development projects*

Growth is presenting us with many challenges. But we can't stop too long and think about how to handle it. We have to continue executing, as we scale up. The problem is like changing the tires on a moving car. It's not easy.

We've been executing since our first day in the garage. As noted above, execution is whenever we meet demand with supply. An idea for a new feature is demand. The time we spend implementing it is supply. The combination of the two is execution. Sometimes it goes well, sometimes it doesn't. Maintaining a tight feedback loop to continually assess our execution is essential.

As we grow into multiple teams and multiple products, we have more complex execution problems, requiring coordination. The fundamental problem is the "D-word:"

dependency. Dependencies are why we coordinate (work with no dependencies can scale nicely along the AKF x-axis.) But when we have dependencies (and there are various kinds) we need a wider range of techniques. Our one Kanban board is not sufficient to the task.

We need to consider the delivery models, as well (the "3 Ps": product, project, process, and now we've added program management.) Decades of industry practice mean that people will tend to think in terms of these models and unless we are clear in our discussions about the nature of our work we can easily get pulled into non-value-adding arguments. To help our understanding, we'll take a deeper look at process management, continuous improvement, and their challenges.

Instructor's note on learning progression

The structure of Part III may be counter-intuitive. Usually, we think in terms of "plan, then execute." However, this can lead to waterfall and deterministic assumptions. Starting the discussion with execution reflects the fact that a scaling company does not have time to "stop and plan." Rather, planning emerges on top of the ongoing execution of the firm, in the interest of controlling and directing that execution across broader time frames and larger scopes of work.

Chapter overview

In this section, we will cover:

- Defining coordination
  - Coordination & dependencies
  - Concepts and techniques
  - Coordination effectiveness
- Coordination, execution, and the delivery models
  - Product management and coordination
  - Project management as coordination
  - Process management as coordination
- A deeper examination of process management
- Process control and continuous improvement

There is a discussion of business process modeling fundamentals in the appendix.

Chapter learning objectives

- Identify and describe dependencies, coordination, and their relationship
- Describe the relationship of delivery models to coordination
- Describe process management and its strengths and weaknesses as a coordination mechanism
- Identify the problems of process proliferation with respect to execution and demand
- Identify key individuals and themes in the history of continuous improvement
- Describe the applicability of statistical process control to different kinds of processes

Defining coordination

Example: Scaling one product

> Good team structure can go a long way toward reducing dependencies but will not eliminate them.
>
> — Mike Cohn *Succeeding with Agile*

> What's typically underestimated is the complexity and indivisibility of many large-scale coordination tasks.
>
> — Gary Hamel *preface to the Open Organization: Igniting Passion and Performance*

We've defined execution as the point at which supply and demand are combined, and of course we've been executing since the start of our journey. Now, however, we are executing in a more complex environment; we have started to scale along the AKF scaling cube y-axis, and we have either multiple teams working on one product, and/or multiple products.Execution becomes more than just "pull another story off the Kanban board." As multiple teams are formed (see Figure 7.1), dependencies arise, and we need coordination. The term "architecture" is likely emerging through these discussions. (We will discuss organizational structure directly in Chapter 9, and architecture in Chapter 12.)

As a consequence of scaling, we are introducing multiple teams as our product scales up (see figure). As noted in the discussion of AmazonAmazon's product strategy, some needs for coordination may be mitigated through the design of the product itself. This is why APIs and microservices are popular architecture styles. If the features and components have well defined protocols for their interaction and clear contracts for matters like performance, development on each team can move forward with some autonomy.

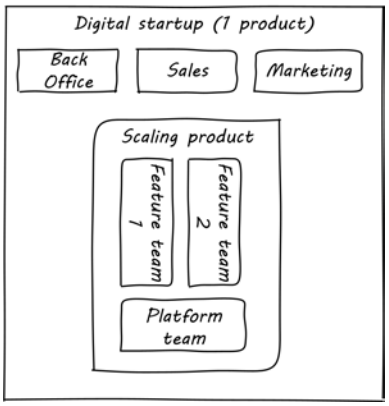


Figure 7.1: Multiple feature teams, one product

But at scale, complexity is inevitable. What happens when a given business objective requires coordinated effort across multiple teams? (Figure.) For example, an online e-commerce site might find itself overwhelmed by business success. Upgrading the site to accommodate the new demand might require distinct development work to be performed by multiple teams (see Figure 7.2).

As the quote from Gary Hamel above indicates, a central point of coordination and accountability is advisable, otherwise the objective is at risk. (It becomes "someone else's problem.") We will return to the investment and organizational aspects of multi-team and multi-product scaling in Chapters 8 and 9. For now, we will focus on dependencies and operational coordination.

A deeper look at dependencies

> . . . coordination can be seen as the process of managing dependencies among activities.
>
> — Malone and Crowston

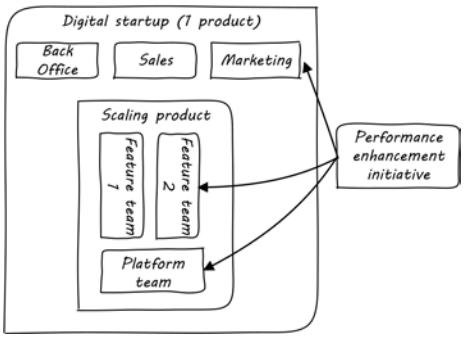


Figure 7.2: Coordinated initiative across timeframes

What is a "dependency"? We need to think carefully about this. According to the definition above (from [169]), without dependencies, we do not need coordination. (We'll look at other definitions of coordination in the next two chapters.) Diane Strode and her associates ([248]) have described a comprehensive framework for thinking about dependencies and coordination, including a dependency taxonomy, an inventory of coordination strategies, and an examination of coordination effectiveness criteria.

To understand dependencies, Strode et al. [249] propose the framework shown in Table 7.1 [59].

We can see examples of these dependencies throughout digital products. In the next section, we will talk about coordination techniques to manage across dependencies.

Part IV

Enterprise

Scenario

You are now running one of the larger and more complex IT-based operations on the planet, with an annual IT budget of hundreds of millions or billions of dollars. You have thousands of programmers, systems engineers, and IT managers, with a wide variety of responsibilities. IT is in your market-facing products and in your back-office operations. In fact, it's sometimes hard to distinguish the boundaries as your company transforms into a digital business.

Agile techniques remain important to you, but things are getting complex and you're testing the boundaries of what is possible. How can you operate at the scale you've achieved and still be Agile? As usual in life, you're finding that there are always tradeoffs. Decisions you made long ago come back to haunt you, security threats are increasing, and at your scale there's no escaping the auditors.

You have scaled up in terms of size, what is less often understood that scaling up in size also means scaling up in terms of timeframes: concern for the past and the future extend further and further in each direction. Organizational history is an increasing factor, and the need to manage this knowledge base can't be ignored.

But you have great resources at your command, and you're as well positioned as any of your competitors to meet the challenges ahead. And in the end, that's all you need.

Chapter 10: Governance, Risk, Security, and Compliance

We need to cope with new layers of enterprise organization, and external forces (regulators, vendor partners, security adversaries, auditors) increasingly defining our options. This chapter sets the frame for the section. Chapters 11 and 12 in many ways are further elaborations of two major domains of governance concerns.

Chapter 11: Enterprise Information Management

We've been concerned with data, information, and knowledge since the earliest days of our journey. But at this scale, we have to formalize our approaches and understandings; without that, we will never capture the full value available with modern analytics and Big Data.

Looking inward, we need to measure this massive IT estate and understand it as an overall dynamic and complex system.

Chapter 12: Architecture and Portfolio

We need to understand the big picture of interacting lifecycles, reduce technical debt and redundancy, and obtain better economies of scale. We need to define our investment strategy based on a sound understanding of both business needs and technology limitations.

Important
Part IV, like the other parts, needs to be understood as a unified whole. In reality, enterprises struggle with the issues in all three chapters simultaneously.

Special section: The IT lifecycles

We've discussed products and the various ways digital organizations deliver them, from simple work management to more sophisticated project and process management approaches. Now, we need to refine our understanding of the products themselves and how they are managed.

We previously discussed the relationship between feature versus component teams in chapter 4. In chapter 9, we touched on the idea of shared services teams. Both of these ideas are now expanded into what is called the "four lifecycle model."

The four lifecycle model was first documented in [22]. The four lifecycles are:

- The application service lifecycle
- The infrastructure service lifecycle
- The asset lifecycle
- The technology product lifecycle

Each of these lifecycles reflects the existence of a significant concept, that is managed over time, as a portfolio. (More on IT portfolio management practices in Chapter 10.)

First, bear in mind that services are *kinds* of products. Digital value is usually delivered as a service, and therefore shares standard service characteristics from an academic perspective, including the idea that services are produced and consumed simultaneously (e.g. an account lookup) and are perishable (a computer's idle time cannot be recovered if it goes unused).

The first two concepts (application and infrastructure service) below reflect these characteristics; the second two (asset and technology product) do not.

An **application service** is a business or market-facing product, consumed by people whose primary activities are **not** defined by an interest in information technology: for example, a bank customer lookiing up her account balance, or an Accounts Payable systems operator. In terms of "feature versus component," the concept of Application is more aligned to "feature." An example would include an Online Banking system or a Payroll system.

The **application service lifecycle** is the end to end existence of such a systems, from idea to retirement. In general, the realization such a system is needed originates **externally** to the IT capability (regardless of its degree of centralization.) Software as a Service usage is also tracked here.

An **infrastructure service** is, by contrast and as previously discussed, a digital or IT service primarily of interest to other digital or IT services/products. Its lifecycle is similar to that of the application service, except that the user is some other IT service. An example would be a storage area network system managed as a service, or the integrated networking system required for connectivity in a data center. Platform and infrastructure as a service is also tracked here.

Note that in terms of our service definition discussion, the above lifecycle concepts are service **systems**. The lifecycle of service offerings is a business lifecycle having more to do with go to market strategy on the part of the firm. We covered this to some extent in Chapter 4 and will revisit it in Chapter 12.

An **asset** is a valuable, tangible investment of organizational resources that is tracked against loss or misuse, and optimized for value over time. It can sit unused and still have some value. Examples would include a physical server or other device, or a commercial software license. Whether assets can be virtual is a subject of debate and specific to the organization's management objectives (Given the licensing implications of virtual servers, treating them as assets is not uncommon.)

The **asset lifecycle** is distinct from the service lifecycles, following a rough order including standard supply chain activities:

- Forecast
- Requisition
- Request quote
- Order
- Deliver
- Accept
- Install/configure
- Operate
- Dispose

A contract reserving Cloud capacity is also an Asset.

Finally, a **technology product** is a **class** of Assets, the "type" to the Asset "instance." For example, the enterprise might select the Oracle relational database as a standard Technology Product. It might then purchase 10 licenses, which are Assets.

The **technology product lifecycle** is also distinct from both the Service and Asset lifecycles:

- Identify technical requirement or need
- Evaluate options

- Select product (may kick off Asset Lifecycle)
- Specify acceptable use
- Maintain vendor relationship
- Maintain product (e.g. patching and version upgrades)
- Continuously evaluate product's fitness for purpose
- Retire product from environment

Cloud services need to be managed in terms of what version they are and what the interoperability concerns are.

The challenge in digital management is "lining up the lifecycles" so that transactional value flows across them (see Figure 9.15 [74]).

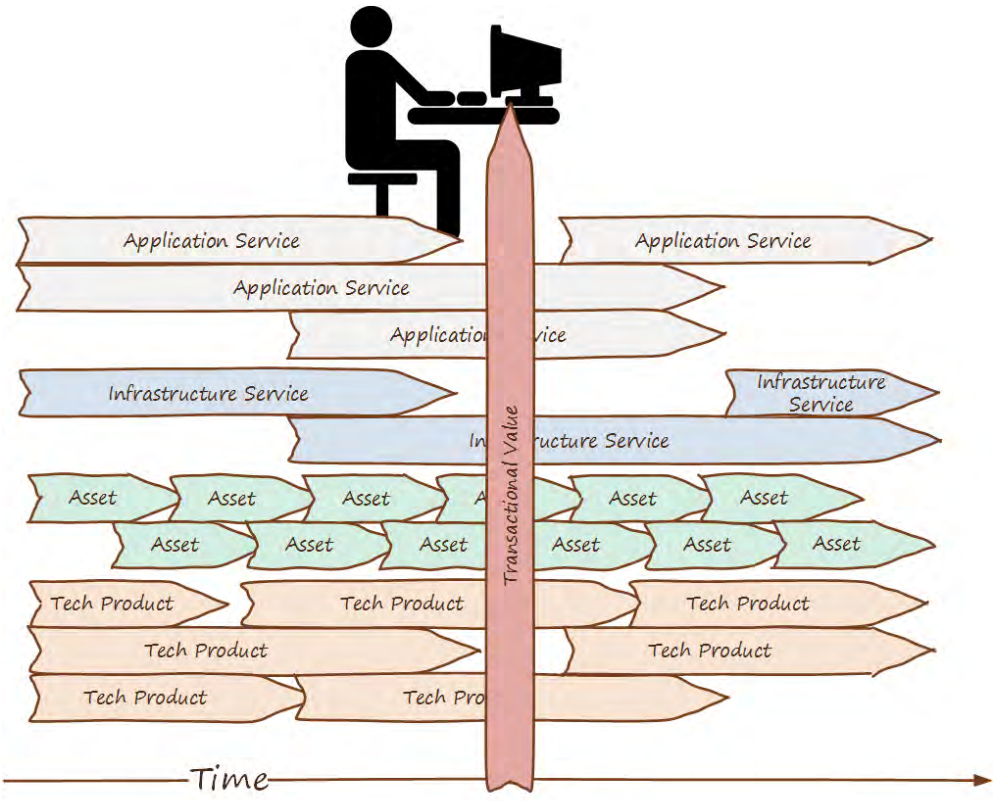


Figure 9.15: Multiple lifecycle model

This can be very difficult, as each lifecycle has a logic of its own, and there may be multiple interdependencies. A technology product may come to the end of its market life and drive expensive changes up the stack. Conversely, new application requirements may expose deficiencies in the underlying stack, again requiring expensive

remediation. Technology product vulnerabilities can prove disruptive, and the asset lifecycle (representing either physical depreciation and refresh cycles, or time-bound licensing) is a significant cost driver.

These lifecycles are essential to all of the next 3 chapters.

- In Chapter 10, they represent focal points for risk and control
- In Chapter 11, we will see how enterprise information management depends on their management
- In Chapter 12, we will further discuss how they are managed in terms of architecture and portfolio.

Chapter 10

Governance, Risk, Security, and Compliance

Introduction

Operating at scale requires a different mindset. When you were starting out, the horizon seemed bounded only by your imagination, will, and talent. At enterprise scale, it's a different world. You find yourself constrained by indifferent forces and hostile adversaries, some of them competing fairly, and others seeking to attack by any means. Whether or not you are a for-profit, publicly traded company, you are now large enough that audits are required; you also likely have directors of some nature. Like it or not, the concept of "controls" has entered your awareness.

As a team of teams, you needed to understand resource management, finance, the basics of multiple product management and coordination, and cross-functional processes. Now that you are an enterprise, you need also to consider questions of corporate governance. Your stakeholders have become more numerous and their demands have multiplied, so the well-established practice of establishing a governing body has been applied.

Security threats increase proportionally to the company's size. The talent and persistence of these adversaries is remarkable. Other challenging players are, on paper, "on the same side," but auditors are never to be taken for granted. Why are they investigating IT systems? What are their motivations and responsibilities? Finally, what laws and regulations are relevant to IT?

Important
As with other chapters in the later part of this book, we are going to some degree introduce this topic "on its own terms." We will then add additional context and critique in subsequent sections.

More than any other chapter, the location of this chapter (and especially its Security subsection) in Section 4 draws attention. Again, any topic in any chapter may be a matter of concern at any stage in an organization's evolution.

You've been doing security since your company started, otherwise you would not have gotten this big. But now, you have a Chief Information Security Officer, formal risk management processes, a standing director-level security steering committee, auditors, and compliance specialists. That kind of formalization does not usually happen until an organization grows to a certain size.

We needed the content in Section 3 to get this far. We had to understand our structure, how we were organizing our strategic investments, and how we were engaging in operational activities. In particular **it's difficult for an organization to govern itself without some ability to define and execute processes, as processes often support governance controls and security protocols.**

This chapter covers "Governance, Risk, Security, and Compliance" because there are clear relationships between these concerns. They have important dimensions of independence as well. It is interesting that Shon Harris' popular Guide to the CISSP starts its discussion of security with a chapter titled "Information Security Governance and Risk Management." Governance leads to a concern for risk, and security specializes in certain important classes of risk. Security requires grounding in governance and risk management.

Compliance is also related but again distinct, as the concern for adherence to laws and regulations, and secondarily internal policy.

Chapter 10 outline

- Governance
- Enablers
- Risk management
- Compliance
- Assurance and audit
- Security
- Digital Governance

Chapter 10 learning objectives

- Define governance versus management
- Describe key objectives of governance according to major frameworks
- Define risk management and its components
- Describe and distinguish assurance and audit, and describe their importance to digital operations
- Discuss digital security concerns and practices

- Identify common regulatory compliance issues
- Describe how governance is retaining its core concerns while evolving in light of digital transformation
- Describe automation techniques relevant to supporting governance objectives throughout the digital delivery pipeline

Governance

What is governance?

> The system by which organizations are directed and controlled.
>
> — Cadbury Report

> The COBIT 5 framework makes a clear distinction between governance and management. These two disciplines encompass different types of activities, require different organisational structures and serve different purposes . . . In most enterprises, governance is the responsibility of the board of directors under the leadership of the chairperson [while] management is the responsibility of the executive management under the leadership of the CEO.
>
> — COBIT 5 Framework *ISACA*

To talk about digital or IT governance, we must talk about governance in general. Governance is a challenging and often misunderstood concept. First and foremost, it must be distinguished from "management." This is not always easy, but remains essential.

A governance example

Here is simple explanation of governance:

Suppose you own a small retail store. For years, you were the primary operator. You might have hired an occasional cashier, but that person had limited authority; they had the keys to the store and cash register, but not the safe combination, nor was their name on the bank account. They did not talk to your suppliers. They received an hourly wage and you gave them direct and ongoing supervision. [75] In this case, you were a manager. Governance was not part of the relationship.

Now, you wish to go on an extended vacation - perhaps a cruise around the world, or a trek in the Himalayas. You need someone who can count the cash and deposit it, and place orders with and pay your suppliers. You need to hire a professional manager (see Figure 10.1 [76]).

They will likely draw a salary, perhaps some percentage of your proceeds, and you will not supervise them in detail as you did the cashier. Instead, you will set overall guidance and expectations for the results they produce. How do you do this? And perhaps even more importantly, how do you trust this person?

Figure 10.1: Someone to "mind the store"

Now, you need governance.

As we see in the above quote, one of the most firmly reinforced concepts in the COBIT guidance (more on this and ISACA in the next section) is the need to distinguish governance from management. Governance is by definition a board-level concern. Management is the CEO's concern. In this distinction, we can still see the shop owner and his or her delegate.

Important

There is too often a tendency to lump all of "management" in with governance. Sometimes it may be said that the VP of Sales, or Human Resources, "governs" their function, for example. While tempting to executives who want to elevate their status, this is not the intent of the term, as we will detail below.

Some theory of goverance

In political science and economics, the need for governance is seen as an example of the principal-agent problem [82]. Our shopkeeper example illustrates this. The hired manager is the "agent," acting on behalf of the shop owner, who is the "principal."

In principal-agent theory, the agent may have different interests than the principal. The agent also has much more information (think of the manager running the shop day to day, versus the owner off climbing mountains). The agent is in a position to do economic harm to the principal; to shirk duty, to steal, to take kickbacks from suppliers. Mitigating such conflicts of interest is a part of governance.

In larger organizations (such as you are now), it's not just a simple matter of one clear owner vesting power in one clear agent. The corporation may be publicly owned, or

in the case of a non-profit, it may be seeking to represent a diffuse set of interests (e.g. environmental issues). In such cases, a group of individuals (directors) is formed, often termed a "board," with ultimate authority to speak for the organization.

The principal-agent problem can be seen at smaller scale within the organization. Any manager encounters it to some degree, in specifying activities or outcomes for subordinates. But this does not mean that the manager is doing "governance," as governance is by definition an organization-level concern.

The fundamental purpose of boards of directors and similar bodies is to take the side of the principal. This is easier said than done; boards can become overly close to an organization's senior management - the senior managers are real people, while the "principal" may be an amorphous, distant body of shareholders and/or stakeholders.

Because governance is the principal's concern, and because the directors represent the principal, governance, including IT governance, is a board-level concern.

There are various principles of corporate governance we will not go into here, such as shareholder rights, stakeholder interests, transparency, and so forth. References on these topics are included in the chapter conclusion (COSO and ISACA are good places to start). However, as we turn to our focus on digital and IT-related governance, there are a few final insights from principal-agent theory that are helpful to understanding governance. Consider:

the heart of principal-agent theory is the trade-off between (a) the cost of measuring behavior and (b) the cost of measuring outcomes and transferring risk to the agent. [82]

What does this mean? Suppose the shopkeeper tells the manager, "I will pay you a salary of $50,000 while I am gone, assuming you can show me you have faithfully executed your daily duties."

The daily duties are specified in a number of checklists, and the manager is expected to fill these out daily and weekly, and for certain tasks, provide evidence they were performed (e.g. bank deposit slips, checks written to pay bills, photos of cleaning performed, etc.). That is a behavior-driven approach to governance. The manager need not worry if business falls off; they will get their money. The owner has a higher level of uncertainty; the manager might falsify records, or engage in poor customer service so that business is driven away. A fundamental conflict of interest is present; the owner wants their business sustained, while the manager just wants to put in the minimum effort to collect the $50,000. When agent responsibilities can be well specified in this manner, it is said they are highly *programmable*.

Now, consider the alternative. Instead of this very scripted set of expectations, the shopkeeper might tell the manager, "I will pay you 50% of the shop's gross earnings, whether they may be. I'll leave you to follow my processes however you see fit. I expect no customer or vendor complaints when I get back."

In this case, the manager's behavior is more aligned with the owner's goals. If they serve customers well, they will likely earn more. There are any number of hard-to-specify behaviors (less *programmable*) that might be highly beneficial.

For example, suppose the store manager learns of an upcoming street festival, a new one that the owner did not know of or plan for. If the agent is managed in terms of their behavior, they may do nothing — it's just extra work. If they are measured in terms of their outcomes, however, they may well make the extra effort to order merchandise desirable to the street fair participants, and perhaps hire a temporary cashier to staff an outdoor booth, as this will boost store revenue and therefore their pay.

(Note that we have considered similar themes in our discussion of Agile and contract management, in terms of risk sharing.)

In general, it may seem that an outcome-based relationship would always be preferable. There is, however, an important downside. It transfers risk to the agent (e.g. the manager). And because the agent is assuming more risk, they will (in a fair market) demand more compensation. The owner may find themselves paying $60,000 for the manager's services, for the same level of sales, because the manager also had to "price in" the possibility of poor sales and the risk that they would only make $35,000.

Finally, there is a way to align interests around outcomes without going fully to performance-based pay. If the manager for cultural reasons sees their interests as aligned, this may mitigate the principal-agent problem. In our example, suppose the store is in a small, tight-knit community with a strong sense of civic pride and familial ties.

Even if the manager is being managed in terms of their behavior, their cultural ties to the community or clan may lead them to see their interests as well aligned with those of the principal. As noted in [82], "Clan control implies goal congruence between people and, therefore, the reduced need to monitor behavior or outcomes. Motivation issues disappear." We have discussed this kind of motivation in Chapter 7, especially in our discussion of control culture and insights drawn from the military.

COSO and control

> Internal control is a process, effected by an entity's board of directors, management, and other personnel, designed to provide reasonable assurance regarding the achievement of objectives relating to operations, reporting, and compliance.
>
> — Committee of Sponsoring Organizations of the Treadway Commission *Internal Control - Integrated Framework*

An important discussion of governance is found in the statements of COSO on the general topic of "control."

Control is a term with broader and narrower meanings in the context of governance. In the area of risk management, "controls" are specific approaches to mitigating risk. However, "control" is also used by COSO in a more general sense to clarify governance.

Bibliography

[1] Martin L. Abbott and Michael T. Fisher. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise (2nd Edition)*. Old Tappan, NJ: Pearson Education, Inc., 2015.

[2] D. Michael Abrashoff. *It's Your Ship: Management Techniques from the Best Damn Ship in the Navy*. 10th Anniv. Grand Central Publishing, 2012. ISBN: 9781455523023. URL: https://www.amazon.com/Its-Your-Ship-Management-Anniversary/dp/145552302X/ref=sr{_}1{_}1?s=books{\&}ie=UTF8{\&}qid=1480362090{\&}sr=1-1{\&}keywords=it{\%}27s+your+ship.

[3] Accounting Coach. *What is cost accounting?* 2016. URL: http://www.accountingcoach.com/blog/what-is-cost-accounting (visited on 05/07/2016).

[4] Gojko Adzic. *Impact Mapping: Making a big impact with software products and projects*. Gojko Adzic, 2012.

[5] Agile Alliance. *Agile Manifesto and Principles*. 2001. URL: http://agilemanifesto.org/principles.html (visited on 01/01/2016).

[6] Agile Alliance. *Subway Map to Agile Practices*. 2016. URL: https://www.agilealliance.org/agile101/subway-map-to-agile-practices/ (visited on 08/11/2016).

[7] Agile Alliance. *Team Definition*. 2015. URL: https://www.agilealliance.org/glossary/team/ (visited on 09/21/2016).

[8] Atsushi Akera. "Edmund Berkeley and the Origins of the ACM". In: *Communications of the ACM* 50.5 (2007), pp. 31–35.

[9] John Allspaw and Paul Hammond. *10 deploys per day: Dev & ops cooperation at Flickr*. San Jose, CA, 2009. URL: http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr.

[10] Scott Ambler. "Agile Outsourcing". In: *Dr. Dobb's Journal* (2005). URL: http://www.drdobbs.com/architecture-and-design/agile-outsourcing/184415344.

[11] Scott W. Ambler and Pramod J Sadalage. *Refactoring databases : evolutionary database design*. Harlow, U.K.: Addison-Wesley, 2006, xxvii, 350 p. ISBN: 0321293533 (hbk.) : ¹35.99. URL: Tableofcontentshttp://www.loc.gov/catdir/toc/ecip063/2005031959.html.

[12] David J Anderson. *Kanban: Successful Evolutionary Change for your Technology Business*. Sequim, WA: Blue Hole Press, 2010.

[13] Mark Andreessen. *Why Software is Eating The World*. 2011. URL: http://www.wsj.com/articles/SB10001424053111903480904576512250915629

[14] Tom Arbogast, Bas Vodde, and Craig Larman. *Agile Contracts Primer*. 2012. URL: http://www.agilecontracts.com/agile{_}contracts{_}primer.pdf.

[15] Joshua Arnold. *Tilt the playing field: discover, nurture, and speed up the delivery of value*. 2013. URL: https://liber.io/v/liberioEpub{_}53aa4d19e3a7e.

[16] Chris Bank and Jerry Cao. *The Guide to Usability Testing*. uxpin.com, 2016. URL: https://www.uxpin.com/studio/ebooks/guide-to-usability-testing/.

[17] Kent Beck. *extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley, 2000, xxi, 190 p. ISBN: 0201616416 (alk. paper).

[18] Steve Bell et al. *Run grow transform : integrating business and lean IT*. Boca Raton, FL: CRC Press, 2013, xlii, 336 p. ISBN: 9781466504493 (alk. paper).

[19] Stefan Bente, Uwe Bombosch, and Shailendra Langade. *Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 Practices*. Waltham, MA: Morgan Kaufman - Elsevier, 2012.

[20] Scott Bernard. *An Introduction to Enterprise Architecture*. AuthorHouse, 2012.

[21] Charles Betz. *Release management integration pattern - seeking devops comments*. 2011. URL: http://www.lean4it.com/2011/01/release-management-integration-pattern-seeking-devops-comments.html.

[22] Charles Betz. *The CMDB is not a data warehouse*. 2011. URL: http://blogs.enterprisemanagement.com/charlesbetz/2012/01/26/cmdb-data-warehouse/.

[23] Charles T Betz. *A DevOps Causal Loop Diagram parts 1 and 2*. 2013. URL: http://www.lean4it.com/2013/05/a-devops-causal-loop-diagram.html (visited on 02/05/2017).

[24] Charles T Betz. *Architecture and Patterns for IT: Service and Portfolio Management and Governance (Making Shoes for the Cobbler's Children), 2nd Edition*. Amsterdam: Elsevier/Morgan Kaufman, 2011.

[25] Charles T Betz. *Calavera Project*. 2016. URL: https://github.com/dm-academy/Calavera (visited on 01/01/2016).

[26] Betsy Beyer et al. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, Inc., 2016.

[27] Randy Bias. *Architectures for open and scalable clouds*. 2012. URL: http://www.slideshare.net/randybias/architectures-for-open-and-scalable-clouds (visited on 11/10/2016).

[28] R.V. Binder. "Can a manufacturing quality model work for software?" In: *IEEE Software* 14.5 (1997), pp. 101–102, 105. ISSN: 07407459. DOI: 10.1109/52.605937. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=605937.

[29] Steve Blank. *The Four Steps to the Epiphany: Successful Strategies for Products That Win*. 2nd. Steve Blank, 2013.

[30] Jason Bloomberg. *Agile Enterprise Architecture Finally Crosses the Chasm*. 2014. URL: http://www.forbes.com/sites/jasonbloomberg/2014/07/23/agile-enterprise-architecture-finally-crosses-the-chasm/print/ (visited on 11/12/2015).

[31] Matt Blumberg. *Startup CEO: A Field Guide to Scaling Up Your Business, + Website*. Wiley, 2013.

[32] Laurent Bossavit. *The Leprechauns of Software Engineering: How folklore turns into fact and what to do about it*. 2015. URL: https://leanpub.com/leprechauns.

[33] Yegevniy Brikman. *Hello, Startup: A Programmer's Guide to Building Products, Technologies, and Teams*. Sebastopol, CA: O'Reilly Media, Inc., 2016.

[34] Frederick P Brooks. *The mythical man-month : essays on software engineering*. Reading, Mass.: Addison-Wesley Pub. Co., 1975, xi, 195 p. ISBN: 0201006502.

[35] Frederick P Brooks. *The mythical man-month : essays on software engineering*. Anniversar. Reading, Mass.: Addison-Wesley Pub. Co., 1995, xiii, 322 p. ISBN: 0201835959.

[36] Alanna Brown et al. *2016 State of DevOps report*. Tech. rep. Puppet Labs, 2016. URL: https://puppet.com/resources/white-paper/2016-state-of-devops-report.

[37] Joshua Brustein. *Microsoft Kills Its Hated Stack Rankings. Does Anyone Do Employee Reviews Right?* 2013. URL: http://www.bloomberg.com/news/articles/2013-11-13/microsoft-kills-its-hated-stack-rankings-dot-does-anyone-do-employee-reviews-right (visited on 03/05/2016).

[38] Werner Buchholz. *Planning a Computer System: Project Stretch*. New York: McGraw-Hill Book Company, Inc., 1962.

[39] Marcus Buckingham and Ashley Goodall. "Reinventing performance management". In: *Harvard Business Review* 93.4 (2015), pp. 40–50. ISSN: 00178012. URL: https://hbr.org/2015/04/reinventing-performance-management.

[40] Mark Bur-gess. *In Search of Certainty*. Sebastopol, CA: O'Reilly Media, Inc., 2015.

[41] Mark Burgess. *When and where order matters*. URL: http://markburgess.org/blog{_}order.html (visited on 11/12/2016).

[42] Roger Burlton. *Business Process Management: Profiting from Process*. Indianapolis, Indiana: SAMS, 2001.

[43] Michael Burrows. *Kanban from the Inside: Understand the Kanban Method, connect it to what you already know, introduce it with impact*. Sequim, Washington: Blue Hole Press, 2015.

[44] Mike; Google Inc. Burrows. "The Chubby lock service for loosely-coupled distributed systems". In: *7th symposium on Operating systems design and implementation (OSDI '06)*. USENIX Association Berkeley, CA, USA ©2006, 2006, Pages 335–350.

[45] Business Architecture Guild. *A Guide to the Business Architecture Body of Knowledge (BIZBOK Guide)*. Business Architecture Guild, 2016.

[46] Brandon Butler. *Cloud's worst-case scenario: What to do if your provider goes belly up*. 2014. URL: http://www.networkworld.com/article/2173255/cloud-computing/cloud-s-worst-case-scenario-what-to-do-if-your-provider-goes-belly-up.html (visited on 07/04/2016).

[47] Brandon Butler. *Free cloud storage service MegaCloud goes dark*. 2013. DOI: http://www.networkworld.com/article/2171450/cloud-computing/free-cloud-storage-service-megacloud-goes-dark.html. (Visited on 07/04/2016).

[48] Marty Cagan. *Inspired: How to Create Products Customers Love*. SVPG Press, 2008. URL: http://www.amazon.com/Inspired-Create-Products-Customers-Love/dp/0981690408.

[49] Murray Cantor. "Agile Management". In: *Cutter IT Journal* (2016). URL: http://www.murraycantor.com/wp-content/uploads/2016/12/Agile-Management-Part-12-MCFINAL.pdf.

[50] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. San Diego: Academic Press, 1999.

[51] Nicholas Carr. "IT Doesn't Matter". In: *Harvard Business Review* (2003), pp. 5–12.

[52] Joe Castaldo. *The Last Days of Target: The untold tale of Target Canada's difficult birth, tough life and brutal death*. 2016. URL: http://www.canadianbusiness.com/the-last-days-of-target-canada/ (visited on 08/30/2016).

[53] Scott Chacon. *Pro Git*. The expert's voice in software development. Berkeley, CA New York: Apress ; Distributed to the book trade worldwide by Springer-Verlag, 2009, xxi, 265 p. ISBN: 9781430218333 (pbk.) 1430218339 (pbk.) 9781430218340 (ebk.)

[54] Kendra Cherry. *Multitasking: Bad for Your Productivity and Brain Health*. 2016. URL: https://www.verywell.com/multitasking-2795003 (visited on 11/28/2016).

[55] Mauro Cherubini et al. “Let ’ s Go to the Whiteboard: How and Why Software Developers Use Drawings”. In: *CHI 2007 Proceedings* (2007), pp. 557–566. ISSN: 10629432. DOI: 10.1145/1240624.1240714. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/p557-cherubini.pdf.

[56] Janet Choi. *The Science Behind Why Jeff Bezos’s Two-Pizza Team Rule Works*. 2014. URL: http://blog.idonethis.com/two-pizza-team/ (visited on 10/22/2016).

[57] Clayton Christensen, Scott Cook, and Taddy Hall. *What Customers Want from Your Products*. 2006. URL: http://hbswk.hbs.edu/item/what-customers-want-from-your-products (visited on 09/18/2016).

[58] Nicola Clark. *The Airbus saga: Crossed wires and a multibillion-euro delay*. New York, N.Y, 2006. URL: http://www.nytimes.com/2006/12/11/business/worldbusiness/11iht-airbus.3860198.html.

[59] Clayton Christensen Institute. *Jobs to be Done*. 2015. URL: http://www.christenseninstitute.org/key-concepts/jobs-to-be-done/ (visited on 09/18/2016).

[60] Ronald Coase. “The nature of the firm”. In: *Economica* 4 (1937), pp. 386–405.

[61] Alistair Cockburn. *Agile Software Development: The Cooperative Game*. 2nd. Boston, MA: Pearson Education, Inc., 2007.

[62] Alistair Cockburn. *Why Agile Works*. 2012. URL: http://www.slideshare.net/AgileLietuva/agile-and-software-engineering-in-the-21st-century (visited on 01/15/2016).

[63] Mike Cohn. “Agile estimating and planning”. In: *VTT Symposium (Valtion Teknillinen Tutkimuskeskus)*. 241. 2006, pp. 37–39. ISBN: 0018-5345. DOI: none. arXiv: arXiv:1011.1669v3.

[64] Mike Cohn. *Succeeding with Agile: Software Development Using Scrum*. Upper Saddle River, New Jersey: Addison-Wesley, 2010.

[65] Swati Comella-Dorda, Santiago Lohiya and Gerard Speksnijder. *An operating model for company-wide agile development*. 2016. URL: http://www.mckinsey.com/Business-Functions/Business-Technology/Our-Insights/An-operating-model-for-company-wide-agile-development.

[66] Committee on the Financial Aspects of Corporate Governance. *Report of the Committee on the Financial Aspects of Corporate Governance (aka Cadbury Report)*. Tech. rep. London: Gee and Co, Ltd., 1992. DOI: DOI:. URL: http://www.jbs.cam.ac.uk/cadbury/report/.

[67] Dr. Melvin E Conway. *How Do Committees Invent?* 1968. URL: http://www.melconway.com/research/committees.html (visited on 11/25/2016).

[68] Alan Cooper, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design*. 2009. URL: http://www.amazon.com/About-Face-Essentials-Interaction-Design-ebook/dp/B008NC0XR2/.

[69] COSO Commission. *Internal Control — Integrated Framework (2013)*. 2013. URL: https://www.coso.org/Documents/990025P-Executive-Summary-final-may20.pdf (visited on 01/21/2017).

[70] Mihaly Csikszentmihalyi. *Flow : the psychology of optimal experience*. 1st. New York: Harper & Row, 1990, xii, 303 p. ISBN: 0060162538.

[71] Ward Cunningham. *Experience Report: The WyCash Portfolio Management System*. 1992. DOI: 10.1145/157710.157715. URL: http://c2.com/doc/oopsla92.htmlhttp://portal.acm.org/citation.cfm?doid=157710.157715$\backslash$nhttp://portal.acm.org/citation.cfm?doid=157709.157715 (visited on 10/06/2016).

[72] The Data Management Association. *The DAMA Guide to The Data Management Body of Knowledge (DAMA-DMBOK Guide)*. Bradley Beach, NJ: Technics Publications, LLC, 2009. ISBN: 978-0-9771400-8-4.

[73] Antonio De Nicola and Michelle Missikoff. "A Lightweight Methodology for Rapid Ontology Engineering". In: *Communications of the ACM2* 59.3 (2016), pp. 79–86.

[74] Sydney Dekker. *The Field Guide to Understanding Human Error*. Burlington, VT: Ashgate Publishing Limited, 2006. URL: http://www.amazon.com/Field-Guide-Understanding-Human-Error/dp/0754648265.

[75] James DeLuccia. *IT COMPLIANCE AND CONTROLS: Best Practices for Implementation*. Hoboken, N.J.: John Wiley & Sons, Inc., 2008.

[76] James DeLuccia et al. *DevOps Audit Defense Toolkit*. Tech. rep. IT Revolution, 2015. URL: http://bit.ly/DevOpsAuditDoc.

[77] DHS. *Report No. 2006-03, The Use of Commercial Data*. Tech. rep. DHS Data Privacy and Integrity Advisory Committee, 2006.

[78] A E Ditri, John C Shaw, and W Atkins. *Managing the EDP function*. N.Y.: McGraw-Hill, 1971. ISBN: 0070170487.

[79] Peter F Drucker. "Managing for Business Effectiveness". In: *Harvard Business Review* (1963).

[80] Peter F Drucker. *Post-capitalist society*. 1st. New York, NY: HarperBusiness, 1993, 232 p. ISBN: 0887306209.

[81] Paul M Duvall, Steve Matyas, and Andrew Glover. *Continuous integration : improving software quality and reducing risk*. Upper Saddle River, NJ: Addison-Wesley, 2007, xxxiii, 283 p. ISBN: 9780321336385 (pbk. alk. paper) 0321336380 (pbk. alk. paper). URL: http://www.loc.gov/catdir/toc/ecip0714/2007012001.html.

[82] Kathleen M Eisenhardt. “Agency Theory: An Assessment and Review”. In: *Source: The Academy of Management Review Academy of Management Review* 14.1 (1989), pp. 57–74. URL: http://www.jstor.org/stable/258191http://www.jstor.org/stable/258191?seq=1{\{}{\&}{\}}cid=pdf-reference{\{}{\#}{\}}references{\{}{_}{\}}tab{\{}{_}{\}}contentshttp://about.jstor.org/terms.

[83] Rob England. *Plus! The Standard+Case Approach: See Service Response in a New Light*. Mana, New Zealand: Two Hills Ltd., 2013.

[84] Eric Evans. *Domain-driven design : tackling complexity in the heart of software*. Boston ; London: Addison-Wesley, 2004, xxx, 528 p. ISBN: 0321125215 : No price.

[85] Richard P. Feynman. “Cargo Cult Science”. In: *Engineering and Science* 33 (1974), pp. 10–13. ISSN: 1062-7987. DOI: 10.1017/S1062798713000124. arXiv: arXiv:1011.1669v3. URL: http://resolver.caltech.edu/CaltechES:37.7.CargoCult.

[86] Thomas Fisher. *Designing Our Way to a Better World*. Minneapolis, MN: University of Minnesota Press, 2016.

[87] Joseph Flahiff. *How organizational agility will save and destroy your company*. 2016. URL: http://searchcio.techtarget.com/tip/How-organizational-agility-will-save-and-destroy-your-company.

[88] Nicole Forsgren. *Continuous Delivery + DevOps = Awesome*. 2016. URL: http://www.slideshare.net/nicolefv/nf-final-agileindia2016.

[89] Nicole Forsgren et al. *2014 State of DevOps Report*. Tech. rep. Puppet Labs, 2016. URL: http://puppetlabs.com/2014-devops-report.

[90] Martin Fowler. *bliki: StranglerApplication*. 2004. URL: http://martinfowler.com/bliki/StranglerApplication.htmlhttp://www.martinfowler.com/bliki/.

[91] Martin Fowler. *BoundedContext*. 2014. URL: http://martinfowler.com/bliki/BoundedContext.html (visited on 09/01/2016).

[92] Martin Fowler. *Is Design Dead?* 2004. URL: http://martinfowler.com/articles/designDead.html.

[93] Martin Fowler. *Patterns of enterprise application architecture*. Boston: Addison-Wesley, 2003, pp. xxiv, 533. ISBN: 0321127420 (alk. paper).

[94] Martin Fowler. *Shu-Ha-Ri*. 2006. URL: http://martinfowler.com/bliki/ShuHaRi.html (visited on 09/21/2016).

[95] Armando Fox, Eric A Brewer, and Armando Fox. *Harvest, Yield and Scalable Tolerant Systems*. 1999.

[96] Paul Furr and Nathan Ahlstrom. *Nail It then Scale It: The Entrepreneur's Guide to Creating and Managing Breakthrough Innovation*. NISI Publishing., 2013.

[97] John Gall. *The Systems Bible: The beginner's guide to systems large and small*. General Systemantics Pr/Liberty, 2012. ISBN: 978-0961825171.

[98] Erich Gamma et al. *Design patterns : elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley, 1995, pp. xv, 395. ISBN: 0201633612 (acid-free paper).

[99] Atul Gawande. *The Checklist Manifesto*. New York, N.Y: Picador, 2010.

[100] Rachel Gillett. *Productivity Hack Of The Week: The Two Pizza Approach To Productive Teamwork — Fast Company — Business + Innovation*. 2014. URL: https://www.fastcompany.com/3037542/productivity-hack-of-the-week-the-two-pizza-approach-to-productive-teamwork (visited on 12/09/2016).

[101] Robert L Glass. *Software runaways*. Upper Saddle River, NJ: Prentice Hall PTR, 1998, pp. xvi, 259. ISBN: 013673443X.

[102] Eliyahu M Goldratt. *Critical chain*. Great Barrington, Ma.: North River, 1997, 246 p. ISBN: 0884271536 (pbk.) : No price.

[103] Eliyahu M Goldratt and Jeff Cox. *The goal : a process of ongoing improvement*. 3rd rev. Great Barrington, MA: North River Press, 2004, 384 p. ISBN: 0884271781.

[104] Bill Goodwin. *How CIOs can raise their 'IT clock speed' as pressure to innovate grows*. 2015. URL: http://www.computerweekly.com/feature/How-CIOs-can-ramp-up-their-IT-clock-speed-as-pressure-grows.

[105] Jeff Gothelf and Josh Seiden Seiden. *Lean UX: Applying Lean Principles to Improve User Experience*. Sebastopol, CA: O'Reilly Media, Inc., 2013.

[106] Michael Griffin. *How To Write a Policy Manual*. www.templatezone.com, 2016. URL: http://www.templatezone.com/download-free-ebook/office-policy-manual-reference-guide.pdf.

[107] Gary Gruver, Mike Young, and Pat Fulghum. *A Practical Approach to Large-Scale Agile Development: How HP Transformed Laserjet Futuresmart Firmware*. Upper Saddle River, N.J.: Pearson Education, Inc., 2013, xxiv, 183 pages. ISBN: 9780321821720 (pbk. alk. paper) 0321821726 (pbk. alk. paper).

[108] Paul Hammant. *Legacy Application Strangulation : Case Studies*. 2013. URL: http://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/http://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/ (visited on 10/23/2016).

[109] Michael Hammer and James Champy. *Reengineering the corporation*. London: Nicholas Brealey, 1993, vi,223p. ISBN: 1857880293 : ¹16.99.

[110] Paul Harmon. *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes*. Amsterdam: Elsevier, 2003.

[111] Verne Harnish. *Scaling Up: How a Few Companies Make It...and Why the Rest Don't*. Gazelles, Inc., 2014.

[112] Patricia Harpring. *Introduction to Controlled Vocabularies: Terminology for Art, Architecture and other Cultural Works*. Los Angeles, CA: Getty Publications, 2010. URL: http://www.getty.edu/research/publications/electronic{_}publications/intro{_}controlled{_}vocab/index.html.

[113] Shon Harris. *CISSP Exam Guide*. 6th. New York: McGraw-Hill Education, 2013.

[114] David C Hay. *Data model patterns : conventions of thought*. New York: Dorset House ; Chichester : Wiley [distributor], 1996, xix,268p. ISBN: 0932633293.

[115] Martha Heller. *GE's Jim Fowler on the CIO role in the digital industrial economy*. 2016. URL: http://www.cio.com/article/3048805/leadership-management/ges-jim-fowler-on-the-cio-role-in-the-digital-industrial-economy.html (visited on 04/18/2016).

[116] Gregor Hohpe and Bobby Woolf. *Enterprise integration patterns : designing, building, and deploying messaging solutions*. Boston ; London: Addison-Wesley, 2003, p. . cm. ISBN: 0321200683 : ¹34.99. URL: http://www.loc.gov/catdir/toc/ecip047/2003017989.html.

[117] Jeremy Hope and Robin Fraser. *Beyond Budgeting Questions and Answers*. Tech. rep. Beyond Budgeting Round Table, 2001. URL: http://bbrt.org/product/bbrt-qa-white-paper-october-2001/.

[118] Michael Housman and Dylan Minor. "Toxic Workers". 2015. URL: http://www.hbs.edu/faculty/PublicationFiles/16-057{_}d45c0b4f-fa19-49de-8f1b-4b12fe054fea.pdf.

[119] Douglas W Hubbard. *How to measure anything : finding the value of "intangibles" in business*. 2nd. Hoboken, N.J.: Wiley, 2010, xv, 304 p. ISBN: 9780470539392 (cloth) 0470539399 (cloth).

[120] Douglas W Hubbard. *The Failure of Risk Management*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2009.

[121] Jez Humble and Joanne Molesky. "Why Enterprises Must Adopt Devops to Enable Continuous Delivery". In: *Cutter IT Journal* 24.8 (2011).

[122] Jez Humble, Joanne Molesky, and Barry O'Reilly. *Lean enterprise*. First edit. 2013, xxi, 317 pages. ISBN: 1449368425 9781449368425.

[123] Watts S Humphrey. *Managing the software process*. Reading, Mass.: Addison-Wesley, 1989, pp. xviii, 494. ISBN: 0201180952.

[124] James R. Huntzinger. *Lean Cost Management: Accounting for Lean by Establishing Flow*. Fort Lauderdale, Fl.: J. Ross Publishing, 2007.

[125] William H Inmon. *Building the Data Warehouse*. Wiley, 1992.

[126] International Auditing and Assurance Standards Board (IAASB). *ISAE 3000 (Revised), Assurance Engagements Other than Audits or Reviews of Historical Financial Information*. Tech. rep. International Federation of Accountants, 2013. URL: https://www.ifac.org/system/files/publications/files/ISAE3000Revised-forIAASB.pdf.

[127] International Institute of Business Analysis (IIBA). *BABOK v3: A Guide to the Business Analysis Body of Knowledge*. Toronto, Canada: International Intitute of Business Analysis, 2015.

[128] Ellen Isaacs and Alan Walendowski. *Designing from both sides of the screen: How Designers and Engineers Can Collaborate to Build Cooperative Technology*. Indianapolis, Indiana: New Riders Publishing, 2002.

[129] ISACA. *COBIT 5 Enabling Information*. Tech. rep. ISACA, 2013.

[130] ISACA. *COBIT 5: Enabling Processes*. 2012, pp. 1–230. ISBN: 9781604202410.

[131] ISACA. *COBIT 5 for Assurance*. Rolling Meadows, IL: ISACA, 2013.

[132] ISACA. *COBIT 5 for Information Security*. Rolling Meadows, IL: ISACA, 2012.

[133] ISACA. *COBIT 5 for Risk*. Ed. by ISACA. Rolling Meadows, IL, 2013.

[134] ISACA. *ITAF: A Professional Practices Framework for IS Audit/ Assurance, 3rd Edition*. Rolling Meadows, IL: ISACA, 2014.

[135] ISO/IEC. *ISO 31000:2009 - Risk Management*. Tech. rep. 2009.

[136] ISO/IEC. *ISO/IEC 38500 - Corporate governance of information technology*. 2008.

[137] Franz Ivancsich, Ralf Kruse, and Dave Sharrock. *Why "Real Options" is the biggest fail of the Agile Community so far*. 2013. URL: http://www.agile42.com/en/blog/2013/04/03/real-options-biggest-fail-agile/ (visited on 12/28/2016).

[138] Stephen H Kan. *Metrics and models in software quality engineering*. Reading, Mass.: Addison-Wesley, 1995, pp. xvii, 344. ISBN: 0201633396 (acid-free paper).

[139] Cem Kaner, Jack L Falk, and Hung Quoc Nguyen. *Testing computer software*. 2nd. New York: Wiley, 1999, pp. xv, 480. ISBN: 0471358460.

[140] Robert S. Kaplan and David P. Norton. "The balanced scorecard - measure that drive performance." In: *Harvard Business Review* January-February (1992), pp. 71–79. ISSN: 00178012. DOI: 00178012. arXiv: 9906032 [cond-mat]. URL: http://www.stevens-tech.edu/MSISCourses/450/Articles/ValueOfIT/TheBalancedScoreCard.pdf.

[141] KARE 11 Staff. *Target cuts 275 positions, most in technology*. 2015. URL: http://www.kare11.com/story/news/2015/09/01/target-cuts-275-jobs-most--technology/71512016/ (visited on 04/17/2016).

[142] Gerold Keefer. *The CMMI Considered Harmful For Quality Improvement And Supplier Selection*. Tech. rep. 2005, p. 26. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.6436{\&}rep=rep1{\&}type=pdf.

[143] Mark Kennaley. *Sdlc 3.0: Beyond a Tacit Understanding of Agile: Towards the Next Generation of Software Engineering*. Fourth Medium Consulting, 2010. ISBN: 0986519405, 9780986519406.

[144] Kevin Kiley. *The Grand Quartier-General Imperial and the Corps d'Armée: Developments in the Military Art, 1795-1815*. 2001. URL: http://www.napoleon-series.org/military/organization/c{_}staff1.html (visited on 10/04/2016).

[145] Gene Kim, Kevin Behr, and George Spafford. *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*. IT Revolution Press, 2013.

[146] Gene Kim et al. *The DevOps Handbook*. Portland, OR: IT Revolution Press, 2016.

[147] Gary Klein, Paul J. Feltovich, and David D. Woods. "Common Ground and Coordination in Joint Activity". In: *Organizational simulation*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2005.

[148] Mark Knez and Duncan Simester. "Making Across-the-Board Incentives Work". In: *Harvard Business Review* Feb 2002 (2002).

[149] Henrik Kniberg. *Culture Over Process*. 2013. URL: https://www.youtube.com/watch?v=Rb0O0Lgs9zU (visited on 10/11/2016).

[150] Henrik Kniberg, Kent Beck, and Kay Keppler. *Lean from the trenches : managing large-scale projects with Kanban*. Dallas, Tex.: Pragmatic Bookshelf, 2011, xiii, 157 p. ISBN: 9781934356852 (pbk.) 1934356859 (pbk.)

[151] Henrik Kniberg and Anders Ivarsson. "Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds". In: (2012). URL: https://dl.dropboxusercontent.com/u/1018963/Articles/SpotifyScaling.pdf.

[152] Ronny Kohavi, Thomas Crook, and Roger Longbotham. *Online Experimentation At Microsoft*. 2009. URL: http://www.exp-platform.com/Documents/ExPThinkWeek2009Public.pdf (visited on 08/31/2016).

[153] Blaze Kos. *Kanban – Visualize your workflow - AgileLeanLife*. 2016. URL: https://agileleanlife.com/kanban-visualize-workflow/ (visited on 12/12/2016).

[154] Gregory Koskela and Lauri Howell. *The underlying theory of project management is obsolete*. 2002. URL: http://www.researchgate.net/publication/3229647{_}The{_}Underlying{_}Theory{_}of{_}Project{_}Management{_}Is{_}Obsolete (visited on 04/01/2016).

[155] C. F. Kurtz and D. J. Snowden. "The new dynamics of strategy: Sense-making in a complex and complicated world". In: *IBM Systems Journal* 42.3 (2003), pp. 462–483. ISSN: 03608581. DOI: 10.1109/EMR.2003.24944.

[156] Corey Ladas. *Scrumban*. Modus Cooperandi Press (January 12, 2009), 2009.

[157] Kin Lane. *The Secret to Amazons Success Internal APIs*. 2012. URL: http://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/ (visited on 12/09/2016).

[158] Douglas Laney. *3D Data Management: Controlling Data Volume, Velocity, and Variety*. Tech. rep. Meta Group (now Gartner), 2001. URL: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[159] Craig Larman and Bas Vodde. *Scaling lean & agile development : thinking and organizational tools for large-scale Scrum*. Upper Saddle River, NJ: Addison-Wesley, 2009, xiv, 348 p. ISBN: 9780321480965 (pbk. alk. paper) 0321480961 (pbk. alk. paper).

[160] Thomas A Limoncelli, Strata R Chalup, and Christina J Hogan. *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems, Volume 2*. Vol. 2. Addison-Wesley Professional, 2014. URL: http://www.amazon.com/Practice-Cloud-System-Administration-Distributed/dp/032194318X/.

[161] Greg Linden. *Early Amazon: Shopping cart recommendations*. 2006. URL: http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html (visited on 11/26/2016).

[162] Sebastian Lins et al. "Dynamic Certification of Cloud Services: Trust, but Verify!" In: *IEEE Security & Privacy* 14.2 (2016), pp. 66–71. ISSN: 1540-7993. DOI: 10.1109/MSP.2016.26. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7448347.

[163] Thomas Lockwood. *Design Thinking: Integrating Innovation, Customer Experience, and Brand Value*. New York, N.Y.: Allworth Press - Allworth Communications, 2010.

[164] Jon Loeliger. *Version control with Git*. 1st. Beijing ; Sebastopol, CA: O'Reilly, 2009, xv, 310 p. ISBN: 9780596520120 (pbk.) 0596520123 (pbk.)

[165] Jack Loftus. *Open source IP case puts spotlight on patents*. 2006. URL: http://searchenterpriselinux.techtarget.com/news/1198160/Open-source-IP-case-puts-spotlight-on-patents (visited on 11/30/2016).

[166] Suzanne Lucas. *Nordstrom's awesome employee handbook is a myth*. 2014. URL: http://www.cbsnews.com/news/nordstroms-awesome-employee-handbook-is-a-myth/ (visited on 06/29/2016).

[167] Raymond J Madachy. *Software process dynamics*. Hoboken, Piscataway, NJ: Wiley IEEE Press, 2008, xxiii, 601 p. ISBN: 9780471274551 (hbk.) URL: http://www.loc.gov/catdir/enhancements/fy0827/2008297228-b.htmlhttp://www.loc.gov/catdir/enhancements/fy0827/2008297228-d.html.

[168] Ruth Malan and Dana Bredemeyer. "The Art of Change: Fractal and Emergent". In: *Cutter Consortium Enterprise Architecture Advisory Service Executive Report* 13.5 (2010).

[169] Thomas W Malone and Kevin Crowston. "The Interdisciplinary Study of Coordination". In: *ACM Computing Surveys* 26.1 (1994).

[170] Howard Marks. *Code Spaces: A Lesson In Cloud Backup*. 2014. URL: http://www.networkcomputing.com/cloud-infrastructure/code-spaces-lesson-cloud-backup/314805651 (visited on 07/04/2016).

[171] David L. Marquet. *Turn the Ship Around!: A True Story of Turning Followers into Leaders: L. David Marquet, Stephen R. Covey: 8601411904479: Amazon.com: Books*. Portfolio, 2013. URL: https://www.amazon.com/Turn-Ship-Around-Turning-Followers/dp/1591846404/ref=sr{_}1{_}1?s=books{\&}ie=UTF8{\&}qid=1480361957{\&}sr=1-1{\&}keywords=turn+the+ship+around.

[172] Chris Matts and Olav Maassen. *"Real Options" Underlie Agile Practices*. 2007. URL: https://www.infoq.com/articles/real-options-enhance-agility (visited on 12/28/2016).

[173] John McAdam. "Information Technology Measurements". In: *Chargeback and IT Cost Accounting*. Ed. by Terence A Quinlan. Santa Barbara, CA: IT Financial Management Association, 2003, pp. 90–91.

[174] Dan McCrory. *Data Gravity – in the Clouds*. 2010. URL: https://blog.mccrory.me/2010/12/07/data-gravity-in-the-clouds/ (visited on 09/01/2016).

[175] N. Dean Meyer. *Internal Market Economics: practical resource-governance processes based on principles we all believe in*. Dansbury, CT: NDMA Publishing, 2013.

[176] Christian Millotat. *Understanding the Prussian-German General Staff System*. Tech. rep. Carlisle Barracks, PA: Strategic Studies Institute (US MIlitary), 1992. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a249255.pdf.

[177] Robert R Moeller. *Executive's Guide to IT Governance: Improving Systems Processes with Service Management, COBIT, and ITIL*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2013.

[178] Dan Moody. "The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering". In: *IEEE Transactions on Software Engineering* 35.5 (2009), pp. 756–778.

[179] Geoffrey Moore. *Crossing the Chasm: Marketing and Selling Disruptive Products to Mainstream Customers*. 3rd. New York, N.Y.: HarperCollins Publishers, Inc., 2014.

[180] JP Morgenthal. *A Reality Check on "Everyone's Moving Everything To The Cloud" — The Tech Evangelist*. 2016. URL: http://jpmorgenthal.com/2016/08/24/a-reality-check-on-everyones-moving-everything-to-the-cloud/ (visited on 11/30/2016).

[181] Kief Morris. *Infrastructure as Code: Managing Servers in the Cloud*. Sebastopol, CA: O'Reilly Media, Inc., 2016.

[182] Randall Munroe. *FedEx Bandwidth*. 2013. URL: http://what-if.xkcd.com/31/ (visited on 09/01/2016).

[183] Jacques Murphy. *Where Should Product Management Report?* 2007. URL: http://pragmaticmarketing.com/resources/where-should-product-management-report.

[184] Sriram Narayam. *Agile IT organization design: for digital transformation and continuous delivery*. Pearson Education Inc., 2015.

[185] Sriram Narayam. *Scaling Agile: Problems and Solutions — ThoughtWorks*. 2015. URL: https://www.thoughtworks.com/insights/blog/scaling-agile-problems-and-solutions?utm{_}campaign=transformation{\&}utm{_}medium=social{\&}utm{_}source=twitter (visited on 11/16/2016).

[186] John von Neumann and Herman H. Goldstine. *Planning and Coding of Problems for an Electronic Computing Instrument*. Tech. rep. Princeton N.J.: Institute for Advanced Study, 1947. URL: https://library.ias.edu/files/pdfs/ecp/planningcodingof0103inst.pdf.

[187] Richard L Nolan. *Managing the data resource function*. 1st ed. West, 1974, 394p.

[188] Inc. Nordstrom. *Code of Business Conduct and Ethics*. 2015. URL: http://investor.nordstrom.com/phoenix.zhtml?c=93295{\&}p=irol-govconduct (visited on 06/29/2016).

[189] Taiichi Ohno. *Toyota production system : beyond large-scale production*. Cambridge, Mass.: Productivity Press, 1988. ISBN: 0915299143.

[190] Elizabeth Olson. "Microsoft, GE, and the futility of ranking employees". In: *Fortune* November 18, 2013 (2013). URL: http://fortune.com/2013/11/18/microsoft-ge-and-the-futility-of-ranking-employees/.

[191] Andreas Opelt et al. *Agile contracts : creating and managing successful projects with Scrum*. 2013, xiv, 282 pages. ISBN: 9781118630945 (paperback) 1118630947 (paperback).

[192] The Open Group. *Archimate 2.1 Specification*. 2012. URL: http://pubs.opengroup.org/architecture/archimate2-doc/toc.html.

[193] The Open Group. *IT4IT Standard*. Tech. rep. G091. 2015. URL: http://www.opengroup.org/it4it/.

[194] The Open Group. *The Open Group Architectural Framework (TOGAF), Version 9.1*. Tech. rep. G091. 2011. URL: http://www.opengroup.org/togaf/.

[195] Alexander Osterwalder and Yves Pigneur. "Business Model Generation". In: *Wiley* (2010), p. 280. ISSN: 47087641. URL: http://www.businessmodelgeneration.com/canvas.

[196] Alexander Osterwalder et al. *Value Proposition Design*. Hoboken, N.J.: John Wiley & Sons, Inc., 2014.

[197] Sydney Padua. *The Thrilling Adventures of Lovelace and Babbage: The (Mostly) True Story of the First Computer*. New York: Random House, 2015.

[198] Jeff Patton. *User Story Mapping. Discover the Whole Story, Build the Right Product*. First edit. 2014, p. 324. ISBN: 1491904887. URL: http://shop.oreilly.com/product/0636920033851.do.

[199] Claude Peck. *U expert tells how 'design thinking' can solve society's big problems*. 2016.

[200] Roman Pichler. *Agile Product Management with Scrum: Creating Products that Customers Love*. Boston, MA: Addison-Wesley - Pearson Education, 2010, p. 160. ISBN: 9780321605788. DOI: 10.1111/j.1540-5885.2011.00829_2.x.

[201] Mary Poppendieck and Thomas David Poppendieck. *Implementing lean software development : from concept to cash*. London: Addison-Wesley, 2007, xxv, 276 p. ISBN: 9780321437389 (pbk.) : ¹28.99 0321437381 (pbk.) : ¹28.99. URL: http://www.loc.gov/catdir/toc/ecip0615/2006019698.html.

[202] Stanley Portny. *Project Management for Dummies*. Hoboken, New Jersey: John Wiley & Sons, 2013.

[203] Derek du Preez. *A CIO's worst nightmare: When your cloud provider goes bankrupt*. 2015. URL: http://diginomica.com/2015/01/06/cios-worst-nightmare-cloud-provider-goes-bankrupt/ (visited on 07/04/2016).

[204] Project Management Institute and Project Management Institute. *A guide to the project management body of knowledge (PMBOK guide)*. 3rd. Newtown Square, Pa.: Project Management Institute Inc., 2013, xxi, 589 pages. ISBN: 9781935589679 (pbk. alk. paper).

[205] Puppet Labs. *2015 State of DevOps Report*. Tech. rep. Puppet Labs, 2015. URL: https://puppet.com/resources/white-paper/2013-state-devops-report.

[206] Terence A Quinlan. *Chargeback and IT Cost Accounting*. Ed. by Terence A Quinlan. Santa Barbara, CA: IT Financial Management Association, 2003.

[207] Bob Raczynski and Bill Curtis. "Software Data Violate SPC's Underlying Assumptions". In: *IEEE Software* 25.3 (2008), pp. 49–51.

[208] Donald J Reifer. *Making the software business case : improvement by the numbers*. Boston: Addison Wesley, 2002, pp. xviii, 300. ISBN: 0201728877.

[209] Donald G Reinertsen. *Managing the design factory: a product developer's toolkit*. New York ; London: Free Press, 1997, xi,269p. ISBN: 0684839911.

[210] Donald G Reinertsen. *The principles of product development flow: second generation lean product development*. Redondo Beach, Calif.: Celeritas, 2009, ix, 294 p. ISBN: 9781935401001 1935401009.

[211] Gary L Richardson. *Project Management Theory and Practice*. Boca Raton: Auerbach Publications, Taylor & Francis Group, 2010.

[212] Eric Ries. *The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses*. 1st. New York: Crown Business, 2011, 320 p. ISBN: 9780307887894 (hbk.) 0307887898 (hbk.)

[213] Darrell K. Rigby, Jeff Sutherland, and Hirotaka Takeuchi. “Embracing Agile”. In: *Harvard Business Review* May (2016). URL: https://hbr.org/2016/05/embracing-agile.

[214] Heidi Rock, David; Grant. *Why Diverse Teams Are Smarter*. 2016. URL: https://hbr.org/2016/11/why-diverse-teams-are-smarter (visited on 11/28/2016).

[215] Everett Rogers. *Diffusion of Innovations*. 5th. New York, N.Y.: Free Press - Simon & Schuster, Inc., 2003.

[216] Jeanne W Ross, Peter Weill, and David Robertson. *Enterprise architecture as strategy : creating a foundation for business execution*. Boston, Mass.: Harvard Business School Press, 2006, xviii, 234 p. ISBN: 1591398398. URL: Tableofcontentshttp://www.loc.gov/catdir/toc/ecip0611/2006010226.html.

[217] Mike Rother. *Toyota kata : managing people for improvement, adaptiveness, and superior results*. New York: McGraw Hill, 2010, xx, 306 p. ISBN: 0071635238 (alk. paper) 9780071635233 (alk. paper).

[218] Mike Rother and John Shook. *Learning to See: Value Stream Mapping to Add Value and Eliminate MUDA [Spiral-bound]*. 2003. DOI: 10.1109/6.490058.

[219] Joanna Rothman. *Not Ready for Agile? Start Your Journey with Release Trains*. 2011. URL: https://www.stickyminds.com/article/not-ready-agile-start-your-journey-release-trains?page=0{\%}2C0 (visited on 12/21/2016).

[220] Walker Royce. “Managing the Development of Large Software Systems”. In: *Proc. IEEE WESCON*. Los Angeles: IEEE, 1970, pp. 1–9.

[221] Julia Rozovsky. “The five keys to a successful Google team”. In: *re:Work* (2015). URL: https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/.

[222] Kenneth S Rubin. *Essential Scrum : a practical guide to the most popular agile process*. Upper Saddle River, NJ: Addison-Wesley, 2012, xliii, 452 p. ISBN: 9780137043293 (pbk. alk. paper) 0137043295 (pbk. alk. paper).

[223] Geary A Rummler and Alan P Brache. *Improving performance: how to manage the white space on the organization chart*. 2nd. San Francisco, CA: Jossey-Bass, 1995, pp. xxv, 226. ISBN: 0787900907 (acid-free paper). URL: http://www.loc.gov/catdir/toc/wiley041/94048105.htmlhttp://www.loc.gov/catdir/description/wiley035/94048105.html.

[224] SAFe. *Agile Release Train – Scaled Agile Framework*. 2016. URL: http://www.scaledagileframework.com/agile-release-train/ (visited on 12/21/2016).

[225] Scaled Agile Framework. *Guidance – Features and Components – Scaled Agile Framework*. 2016. URL: http://www.scaledagileframework.com/features-and-components/ (visited on 11/26/2016).

[226] Steve Schlarman. *Developing Effective Policy, Procedure, and Standards*. 2008. URL: http://www.disaster-resource.com/articles/07p{_}106.shtml (visited on 06/30/2016).

[227] William E. Schneider. *The reengineering alternative : a plan for making your current culture work*. McGraw-Hill, 1999, p. 173. ISBN: 9780071359818.

[228] Ken Schwaber. *Agile Software Development with Scrum*. Upper Saddle River, N.J.: Prentice Hall, 2002.

[229] Ken Schwaber. *The Enterprise and Scrum*. Redmond, Wash: Microsoft Press, 2007.

[230] Ken Schwaber. *unSAFE at any speed*. 2013. URL: https://kenschwaber.wordpress.com/2013/08/06/unsafe-at-any-speed/.

[231] Sean Landis. *Agile Hiring*. Artima, Inc, 2011. ISBN: 978-0981531632.

[232] S B ; Fixott Sells Richard S. and S. B. ; Fixott Sells. "Evaluation of Research on Effects of Visual Training on Visual Functions". In: *Am J Ophthal* 44.2 (1957), pp. 230–236. URL: http://www.ajo.com/article/0002-9394(57)90012-0/abstract.

[233] Claude Elwood Shannon and Warren Weaver. *The mathematical theory of communication*. Urbana, University of Illinois Press, 1949, v (i.e. vii), 117 p.

[234] Anshu Sharma. *Why Big Companies Keep Failing: The Stack Fallacy*. 2015. URL: http://techcrunch.com/2016/01/18/why-big-companies-keep-failing-the-stack-fallacy/ (visited on 04/06/2016).

[235] Alec Sharp and Patrick McDermott. *Workflow modeling : tools for process improvement and applications development*. 2nd. Boston: Artech House, 2009, xx, 449 p. ISBN: 9781596931923 1596931922.

[236] Eric Sigler. *So, What is ChatOps? And How do I Get Started?* 2014. URL: https://www.pagerduty.com/blog/what-is-chatops/ (visited on 06/09/2016).

[237] Len Silverston. *The data model resource book Vol 1: A library of universal data models for all enterprises*. Rev. ed. New York ; Chichester: Wiley, 2001, p. 2 v. ISBN: 0471380237 (pbk.) : ¹35.95.

[238] Len Silverston. *The data model resource book Vol 3: Universal patterns for data modeling*. Indianapolis, Ind.: Wiley, 2008, xxxii, 606 p. ISBN: 9780470178454 (pbk.) : ¹27.99 0470178450 (pbk.) : ¹27.99.

[239] Len Silverston and John Wiley & Sons. *The data model resource CD-ROM. Volume 1 a library of universal data models for all enterprises*. New York, 2001. URL: http://www.loc.gov/catdir/bios/wiley045/2001561468.htmlhttp://www.loc.gov/catdir/description/wiley039/2001561468.html.

[240] Herbert A. Simon. "The Science of Design: Creating the Artificial". In: *Design Issues* 4.1/2 (1988). URL: http://www.jstor.org/stable/1511391.

[241] Chris; Johnson Sims. *Scrum: a Breathtakingly Brief and Agile Introduction*. Dymaxicon, 2012. ISBN: B007P5N8D4.

[242] Rami Sirkiä and Maarit Laanti. *Lean and Agile Financial Planning*. Tech. rep. via Scaled Agile Framework website, 2013. URL: http://www.scaledagileframework.com/original-whitepaper-lean-agile-financial-planning-with-safe/.

[243] Preston G Smith and Donald G Reinertsen. *Developing products in half the time*. New York, N.Y.: Van Nostrand Reinhold, 1991. ISBN: 0442002432.

[244] Preston G Smith and Donald G Reinertsen. *Developing products in half the time : new rules, new tools*. [New ed.] New York ; London: Van Nostrand Reinhold, 1998, xix, 298p. ISBN: 0442025483.

[245] Diomidis Spinellis. "Extending Our Field's Reach". In: *IEEE Software* (2015), pp. 4–6. URL: https://www.computer.org/csdl/mags/so/2015/06/mso2015060004.pdf.

[246] Jim Spohrer et al. "The Service System is the Basic Abstraction of Service Science". In: *41st Hawaii International Conference on System Sciences*. Hawaii, 2008.

[247] John Sterman. *Business dynamics : systems thinking and modeling for a complex world*. Boston: Irwin/McGraw-Hill, 2000, xxvi, 982 p. ISBN: 0072311355 (alk. paper).

[248] Diane E Strode and Sid L Huff. "A Taxonomy of Dependencies in Agile Software Development". In: *23rd Australasian Conference on Information Systems*. 2012.

[249] Diane E Strode et al. "Coordination in co-located agile software development projects". In: *The Journal of Systems and Software* 85 (2012), pp. 1222–1238. URL: https://pdfs.semanticscholar.org/7cc7/2fdff2c46c94fbfaccedb
pdf.

[250] Bjarne Stroustrup. "Viewpoint: What should we teach new software developers? Why?" In: *Communications of the ACM* 53.1 (2010), p. 40. ISSN: 00010782. DOI: 10.1145/1629175.1629192. URL: http://portal.acm.org/citation.cfm?doid=1629175.1629192.

[251] Jeff Sussna. *Designing Delivery: Rethinking IT in the Digital Service Economy*. O'Relly Publications, 2015.

[252] Robert I.; Sutton and Huggy Rao. *Scaling up excellence : getting to more without settling for less*. Crown Business/Random House, 2014.

[253] The National Court Rules Committee. *Federal Rules of Civil Procedure*. 2016. URL: https://www.federalrulesofcivilprocedure.org/.

[254] The Stationery Office. *ITIL Continual Service Improvement: 2011 Edition*. Norwich, U.K.: The Stationery Office, 2011.

[255] The Stationery Office. *ITIL Service Design: 2011 Edition*. Norwich, U.K.: The Stationery Office, 2011.

[256] Jennifer Tidwell. *Designing Interfaces*. Sebastopol, CA: O'Reilly Media, Inc., 2006.

[257] Des Traynor. *Focus on the Job, Not the Customer*. 2016. URL: https://blog.intercom.com/when-personas-fail-you/ (visited on 09/18/2016).

[258] Michael Treacy and Fred Wiersema. *The Discipline of Market Leaders: Choose Your Customers, Narrow Your Focus, Dominate Your Market*. New York, N.Y.: Basic Books - Perseus Books Group, 1997.

[259] Edward R Tufte. *The Visual Display of Quantitative Information*. Vol. 4. 2001. ISBN: 0961392142.

[260] William Ulrich and Neal McWhorter. *Business Architecture: The Art and Practice of Business Transformation*. Tampa, Florida: Meghan-Kiffer, 2010.

[261] Uptime Institute. *Explaining the Uptime Institute's Tier Classification System*. 2014. URL: https://journal.uptimeinstitute.com/explaining-uptime-institutes-tier-classification-system/ (visited on 07/04/2016).

[262] Uptime Institute. *Tier Certification Tiers is the Global Language of Data Center Performance Tier Certification is Worldwide Credibility*. 2016. URL: https://uptimeinstitute.com/uptime{_}assets/2edec7f3207b2802cf5fad3ad50d8594700006.pdf (visited on 07/04/2016).

[263] Stephen L Vargo and Robert F Lusch. "Evolving to a New Dominant Logic for Marketing". In: *Journal of Marketing* 68.January 2004 (2004), pp. 1–17.

[264] Paul Venezia. *Murder in the Amazon cloud*. 2014. URL: http://www.infoworld.com/article/2608076/data-center/murder-in-the-amazon-cloud.html (visited on 07/04/2016).

[265] Archana Venkatraman. *2e2 datacentre administrators hold customers' data to £1m ransom*. 2013. URL: http://www.computerweekly.com/news/2240177744/2e2-datacentre-administrators-hold-customers-data-to-1m-ransom (visited on 07/04/2016).

[266] David Vergun. *Toxic leaders decrease Soldiers' effectiveness, experts say*. 2015. URL: https://www.army.mil/article/157327/Toxic{_}leaders{_}decrease{_}Soldiers{_}{_}effectiveness{_}{_}experts{_}say (visited on 07/13/2016).

[267] Karl E. Weick and Karlene Roberts. "Collective Mind in Organizations: Heedful Interrelating on Flight Decks". In: *Administrative Science Quarterly* 38 (1993), pp. 357–381.

[268] Gerald M. Weinberg. *An introduction to general systems thinking / Gerald M. Weinberg*. Silver ann. New York: Dorset House, 2001, xxi, 279 p. ISBN: 0932633498 (pbk.)

[269] George Westerman, Didier Bonnet, and Andrew Mcafee. "The Nine Elements of Digital Transformation". In: *MIT Sloan Management Review* January (2014), pp. 1–6.

[270] WFMC. *Adaptive Case Management (Thai)*. 2010. URL: http://www.xpdl.org/nugen/p/adaptive-case-management/public.htm (visited on 06/04/2016).

[271] James A Whittaker, Jason Arbon, and Jeff Carollo. *How Google tests software*. Upper Saddle River, NJ: Addison-Wesley, 2012, xxvii, 281 p. ISBN: 9780321803023 (pbk. alk. paper) 0321803027 (pbk. alk. paper).

[272] Wikipedia. *DevOps*. 2016. URL: https://en.wikipedia.org/wiki/DevOps (visited on 05/18/2016).

[273] James P Womack and Daniel T Jones. *Lean thinking: banish waste and create wealth in your corporation*. 1st Free P. New York: Free Press, 2003, 396 p. ISBN: 0743249275. URL: http://www.loc.gov/catdir/bios/simon053/2003279577.htmlhttp://www.loc.gov/catdir/toc/fy042/2003279577.htmlhttp://www.loc.gov/catdir/description/simon041/2003279577.htmlhttp://www.loc.gov/catdir/enhancements/fy0641/2003279577-s.html.

[274] Anita Woolley, Thomas W. Malone, and Christopher F. Chabris. *Why Some Teams Are Smarter Than Others*. 2015. URL: http://www.nytimes.com/2015/01/18/opinion/sunday/why-some-teams-are-smarter-than-others.html?.

[275] Serdar Yegulalp. *Why GPL still gives enterprises the jitters — InfoWorld*. 2014. URL: http://www.infoworld.com/article/2608340/open-source-software/why-gpl-still-gives-enterprises-the-jitters.html (visited on 11/30/2016).

[276] John Zachman. "Zachman Framework". In: *IBM Systems Journal* 26.3 (1987), pp. 276–292.

Glossary

cost of delay the opportunity costs that surface as a function of time, e.g. in project delivery. [209]

Backlog and release notes

LeanPub release Feb 2016 This is the LeanPub early release edition. The book is complete in content but still undergoing polishing. In the same Lean Startup spirit as it attempts to describe, the book is released as a minimum viable product.

Backlog

In publishing, an "errata" section would be appended to books, rather than re-typesetting them. In similar vein, included here is a backlog of work in progress on the book.

- Further editorial
- Professional indexing
- Professional graphics & layout
- Print-targeted PDF version with page numbered cross references
- Ongoing content review by subject matter experts
- Marginal notes
- Glossary

Colophon

- Early ideation in Scrivener.
- GitHub
- AsciiDoc markup
- Asciidoctor rendering (and all related dependencies), for HTML and DocBook output
- Atom text editor
- GIMP
- Microsoft Paint
- Visio
- db2latex for converting DocBook to Latex
- bash and perl scripting to clean up db2latex output & correct various things
- LaTeX and related technologies
- Ubuntu Linux
- Mac OS

Asciidoc toolchain was derived from the Pro Git toolchain. Many images downloaded from Flickr, limited to those tagged commercial use only.

Author biography

Figure 12.45: Charles Betz

Charles Betz, founder of Digital Management Academy, is a specialist in digital and IT management, focusing on IT operating models. He is currently an instructor at the University of St. Thomas and also consults, trains, and advises. He spent 6 years at Wells Fargo as Enterprise Architect for IT Portfolio Management and Systems Management. He has held product owner, architect and analyst positions for AT&T, Best Buy, Target, EMA, and Accenture, specializing in IT management, Cloud, and enterprise architecture.

He is the author of *Architecture and Patterns for IT Management*, a comprehensive reference architecture for the "business of IT." He has served as an ITIL reviewer and COBIT author, and co-author of several works with Lean collaborators and for ISACA's COBIT. Invited to the IT4IT Consortium Strategy Board, Charles led the IT4IT transition to an open standard as The Open Group IT4IT Forum. He is active in the Minnesota professional community, and has served as the chair of the Minnesota Association of Enterprise Architects. Charlie presents and keynotes frequently at national and international events related to IT service management, architecture, and Agile methods. His current interests include:

- IT management and organization
- Agile and DevOps methods, especially in terms of underlying theory and effective pedagogy
- Improving higher education institutional response to digital workforce needs
- IT frameworks and bodies of knowledge (ITIL, TOGAF, COBIT, IT4IT) and their relationship to IT operating models
- Virtual simulations of digital delivery environments for instructional purposes

Charlie lives in Minneapolis, Minnesota with his wife Sue and son Keane.

Index