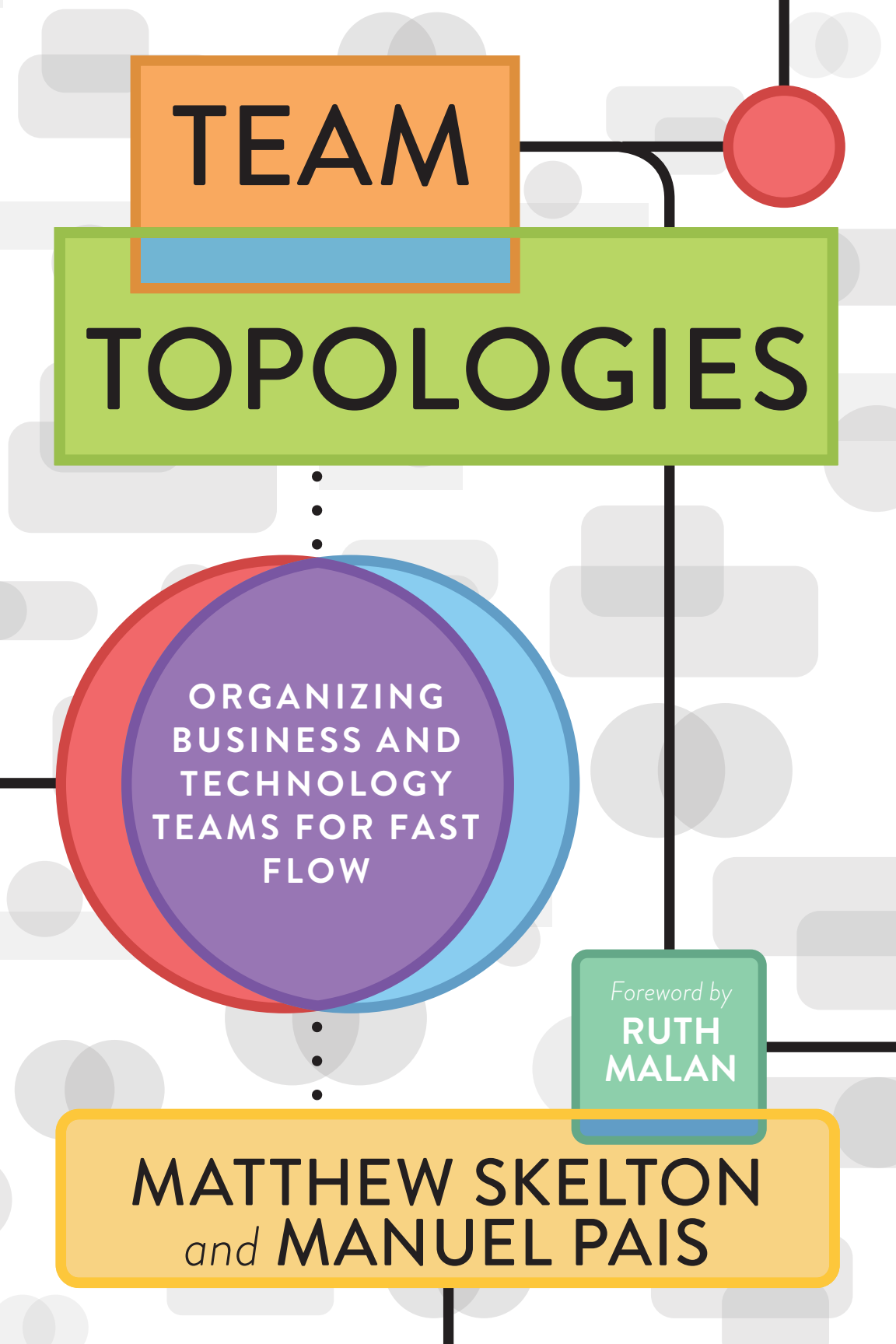


The background of the book cover features a network diagram with black lines and nodes. A red circle is at the top right, and a purple circle is at the bottom center. The text is arranged in a structured layout with colored boxes and a central Venn diagram.

TEAM

TOPOLOGIES

ORGANIZING
BUSINESS AND
TECHNOLOGY
TEAMS FOR FAST
FLOW

Foreword by
**RUTH
MALAN**

MATTHEW SKELTON
and **MANUEL PAIS**

TEAM TOPOLOGIES

ORGANIZING BUSINESS AND
TECHNOLOGY TEAMS
FOR FAST FLOW

MATTHEW SKELTON
and MANUEL PAIS

Foreword by Ruth Malan

IT Revolution
Portland, Oregon



25 NW 23rd Pl, Suite 6314

Portland, OR 97210

Copyright © 2019 by Matthew Skelton and Manuel Pais

All rights reserved. For information about permission to reproduce selections from this book,
write to Permissions, IT Revolution Press, LLC,
25 NW 23rd Pl, Suite 6314, Portland, OR 97210

First Edition

Printed in the United States of America

24 23 22 21 20 19

1 2 3 4 5 6 7 8 9 10

Cover and book design by Devon Smith

Author Photographs by Gary Landsman

Library of Congress Catalog-in-Publication Data

Names: Matthew Skelton (tk) and Manuel Pais (tk), authors.

Title: Team Topologies : organizing business and technology teams
for fast flow / by Matthew Skelton and Manuel Pais.

Description: Portland, OR : IT Revolution Press, [2019] |

Includes bibliographical references.

Identifiers: LCCN 2018047857 | ISBN 9781942788-812 (trade pbk.) |

ISBN 9781942788829 (ePub) | ISBN 9781942788836 (kindle) |

ISBN 9781942788843 (pdf)

Subjects: LCSH: Information technology—Management. |

Technological innovations—Management. | Leadership.

Classification: LCC HD30.2 .S3878 2019 | DDC 004.068—dc23

LC record available at <https://lcn.loc.gov/2018047857>

For information about special discounts for bulk purchases or for information
on booking authors for an event, please visit our website at ITRevolution.com.

Team Topologies

CONTENTS

Figures & Tables	x
Case Studies & Industry Examples	xi
Foreword by Ruth Malan	xv
Preface	xvii

PART I TEAMS AS THE MEANS OF DELIVERY

Chapter 1: The Problem with Org Charts	3
Communication Structures of an Organization	4
Team Topologies: A New Way of Thinking about Teams	9
The Revival of Conway's Law	9
Cognitive Load and Bottlenecks	11
Summary: Rethink Team Structures, Purpose, and Interactions	13
Chapter 2: Conway's Law and Why It Matters	15
Understanding and Using Conway's Law	15
The Reverse Conway Maneuver	18
Software Architectures that Encourage Team-Scoped Flow	21
Organization Design Requires Technical Expertise	23
Restrict Unnecessary Communication	24
Beware: Naive Uses of Conway's Law	26
Summary: Conway's Law Is Critical for Efficient Team Design in Tech	29
Chapter 3: Team-First Thinking	31
Use Small, Long-Lived Teams as the Standard	32
Good Boundaries Minimize Cognitive Load	39
Design "Team APIs" and Facilitate Team Interactions	46
Warning: Engineering Practices Are Foundational	55
Summary: Limit Teams' Cognitive Load and Facilitate Team Interactions to Go Faster	56

PART II TEAM TOPOLOGIES THAT WORK FOR FLOW

Chapter 4: Static Team Topologies	61
Team Anti-Patterns	62
Design for Flow of Change	63
DevOps and the DevOps Topologies	65

Successful Team Patterns	67
Considerations When Choosing a Topology	72
Use DevOps Topologies to Evolve the Organization	75
Summary: Adopt and Evolve Team Topologies that Match Your Current Context	77
Chapter 5: The Four Fundamental Team Topologies	79
Stream-Aligned Teams	81
Enabling Teams	86
Complicated-Subsystem Teams	91
Platform Teams	92
Avoid Team Silos in the Flow of Change	99
A Good Platform Is “Just Big Enough”	100
Convert Common Team Types to the Fundamental Team Topologies	104
Summary: Use Loosely Coupled, Modular Groups of Four Specific Team Types	109
Chapter 6: Choose Team-First Boundaries	111
A Team-First Approach to Software Responsibilities and Boundaries	112
Hidden Monoliths and Coupling	112
Software Boundaries or “Fracture Planes”	115
Real-World Example: Manufacturing	121
Summary: Choose Software Boundaries to Match Team Cognitive Load	123
PART III EVOLVING TEAM INTERACTIONS FOR INNOVATION AND RAPID DELIVERY	
Chapter 7: Team Interaction Modes	131
Well-Defined Interactions Are Key to Effective Teams	132
The Three Essential Team Interaction Modes	133
Team Behaviors for Each Interaction Mode	137
Choosing Suitable Team Interaction Modes	144
Choosing Basic Team Organization	146
Choose Team Interaction Modes to Reduce Uncertainty and Enhance Flow	149
Summary: Three Well-Defined Team Interaction Modes	151

Chapter 8: Evolve Team Structures with Organizational Sensing	153
How Much Collaboration Is Right for Each Team Interaction?	153
Accelerate Learning and Adoption of New Practices	155
Constant Evolution of Team Topologies	159
Combining Teams Topologies for Greater Effectiveness	164
Triggers for Evolution of Team Topologies	165
Self Steer Design and Development	170
Summary: Evolving Team Topologies	175
 Conclusion: The Next-Generation Digital Operating Model	 177
Four Team Types and Three Interaction Modes	178
Team-First Thinking: Cognitive Load, Team API, Team-Sized Architecture	179
Strategic Application of Conway's Law	180
Evolve Organization Design for Adaptability and Sensing	181
Team Topologies Alone Are Not Sufficient for IT Effectiveness	181
Next Steps: How to Get Started with Team Topologies	183
 Glossary	 187
Recommended Reading	189
References	191
Notes	203
Index	207
Acknowledgments	215
About the Authors	216

FIGURES & TABLES

FIGURES

0.1: The Four Team Types and Three Interaction Modes	xx
1.1: Org Chart with Actual Lines of Communication	6
1.2: Obstacles to Fast Flow	12
2.1: Four Teams Working on a Software System	19
2.2: Software Architecture from Four-Team Organization	20
2.3: Microservices Architecture with Independent Services and Data Stores	21
2.4: Team Design for Microservices Architecture with Independent Services and Data Stores	22
2.5: Inter-Team Communication	25
3.1: Scaling Teams Using Dunbar's Number	34
3.2: No More than One Complicated or Complex Domain per Team	44
3.3: Typical vs. Team-First Software Subsystem Boundaries	45
3.4: Office Layout at CDL	52
4.1: Organization not Optimized for Flow of Change	64
4.2: Organization Optimized for Flow of Change	65
4.3: Relationship between SRE Team and Application Team	71
4.4: Influence of Size and Engineering Discipline on Team Interaction Patterns	73
5.1: The Four Fundamental Team Topologies	80
5.2: Platform Composed of Several Fundamental Team Topologies	96
5.3: Traditional Infrastructure Team Organization	105
5.4: Support Teams Aligned to Stream of Change	107
6.1: Mobile, Cloud, and IoT Technology Fracture Plane Scenario	124
7.1: Collaboration vs. X-as-a-Service	133
7.2: The Three Essential Team Interaction Modes	134
7.3: Team Interaction Modes Scenario	135
7.4: X-as-a-Service Team Interaction Mode	138
7.5: Primary Interaction Modes for the Four Fundamental Team Topologies	145
7.6: Team Interaction Modes at IBM around 2014	146
8.1: Collaboration between Cloud and Embedded Teams	156
8.2: System Build and Platform Build Team at TransUnion	158

8.3: System Build and Platform Build Team Collaboration at TransUnion	158
8.4: System Build and Platform Build Teams Merged at TransUnion	158
8.5: System Build and Platform Build Teams Merged Back Into Dev and Ops at TransUnion	159
8.6: Evolution of Team Topologies	160
8.7: Evolution of Team Topologies in an Enterprise	160
8.8: Example of a “Platform Wrapper”	168
8.9: New-Service and “Business as Usual” (BAU) Teams	173
8.10: Side-by-Side New Service and BAU Teams	174
9.1: Core Ideas of Team Topologies	178

TABLES

Table 7.1: Advantages and Disadvantages of Collaboration Mode	137
Table 7.2: Advantages and Disadvantages of X-as-a-Service Mode	140
Table 7.3: Advantages and Disadvantages of Facilitating Mode	141
Table 7.4: Team Interaction Modes of the Fundamental Team Topologies	144

CASE STUDIES & INDUSTRY EXAMPLES

Chapter 1

Industry Example: OutSystems (Part 1)—Miguel Antunes, R&D Principal Software Engineer, OutSystems	11
---	----

Chapter 2

Industry Example: Adidas—Fernando Cornago, Senior Director Platform Engineering, and Markus Rautert, Vice President Platform Engineering and Architecture, Adidas	16
--	----

Chapter 3

Industry Example: OutSystems (Part 2)—Miguel Antunes, R&D Principal Software Engineer, OutSystems	42
Industry Example: IKEA—Albert Bertilsson, Solution Team Lead, and Gustaf Nilsson Kotte, Web Developer, IKEA	46

Case Study: Team-Focused Office Space at CDL— Michael Lambert, Head of Development, and Andy Rubio, Development Team Leader, CDL	50
Case Study: Stream-Aligned Office Layout for Flow-Based Collaboration at Auto Trader— Dave Whyte, Operations Engineering Lead, and Andy Humphrey, Head of Customer Operations, Auto Trader	53
Chapter 4	
Industry Example: Spotify—Henrik Kniberg, Agile/Lean Coach, and Anders Ivarsson, Organizational Coach, Spotify	63
Industry Example: Feature Teams Supported by Cross- Subsystem Functions at Ericsson—Wolfgang John, Research Leader, Ericsson	68
Industry Example: DevOps Team Topologies at a Healthcare Organization—Pulak Agrawal, DevOps Manager and Technology Architect, Accenture	75
Case Study: Evolution of Team Topologies at TransUnion (Part 1)— Ian Watson, Head of DevOps, TransUnion	76
Chapter 5	
Case Study: Strictly Independent Service Teams at Amazon	82
Case Study: Engineering Enablement Team within a Large Legal Organization—Robin Weston, Engineering Leader, BCG Digital Ventures	88
Case Study: Sky Betting & Gaming—Platform Feature Teams (Part 1)— Michael Maibaum, Chief Architect, Sky Betting & Gaming	94
Case Study: Evolving Highly Responsive IT Operations at Auto Trader—Dave Whyte, Operations Engineering Lead, and Andy Humphrey, Head of Customer Operations, Auto Trader	97
Chapter 6	
Case Study: Finding Good Software Boundaries at Poppulo— Stephanie Sheehan, VP of Operations, and Damien Daly, Director of Engineering, Poppulo	121
Chapter 7	
Case Study: Team Interaction Diversity at IBM around 2014— Eric Minick, Program Director for Continuous Delivery, IBM	146

Chapter 8

Case Study: Adoption of Kubernetes to Drive Organizational Change at uSwitch—Paul Ingles, Head of Engineering, uSwitch	155
Case Study: Evolution of Team Topologies at TransUnion (Part 2)—Dave Hotchkiss, Platform Build Manager, TransUnion	157
Case Study: Sky Betting and Gaming—Platform Feature Teams (Part 2)—Michael Maibaum, Chief Architect, Sky Betting & Gaming	162

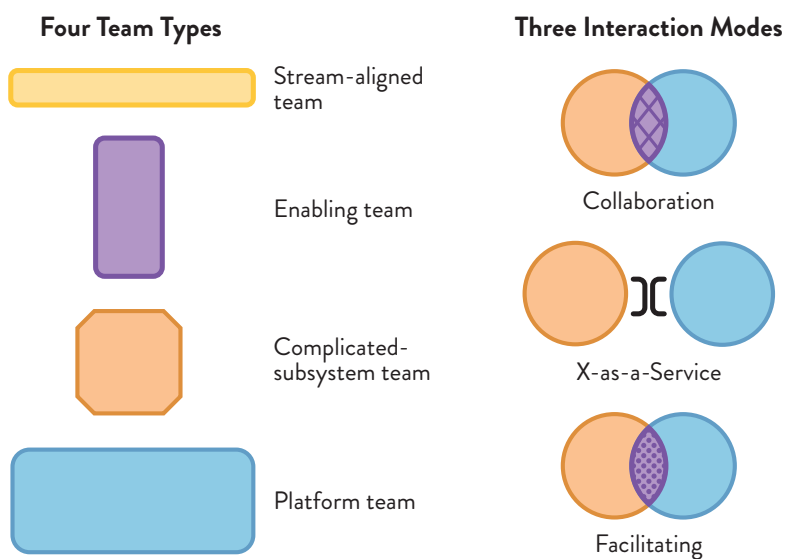


Figure 0.1: The Four Team Types and Three Interaction Modes

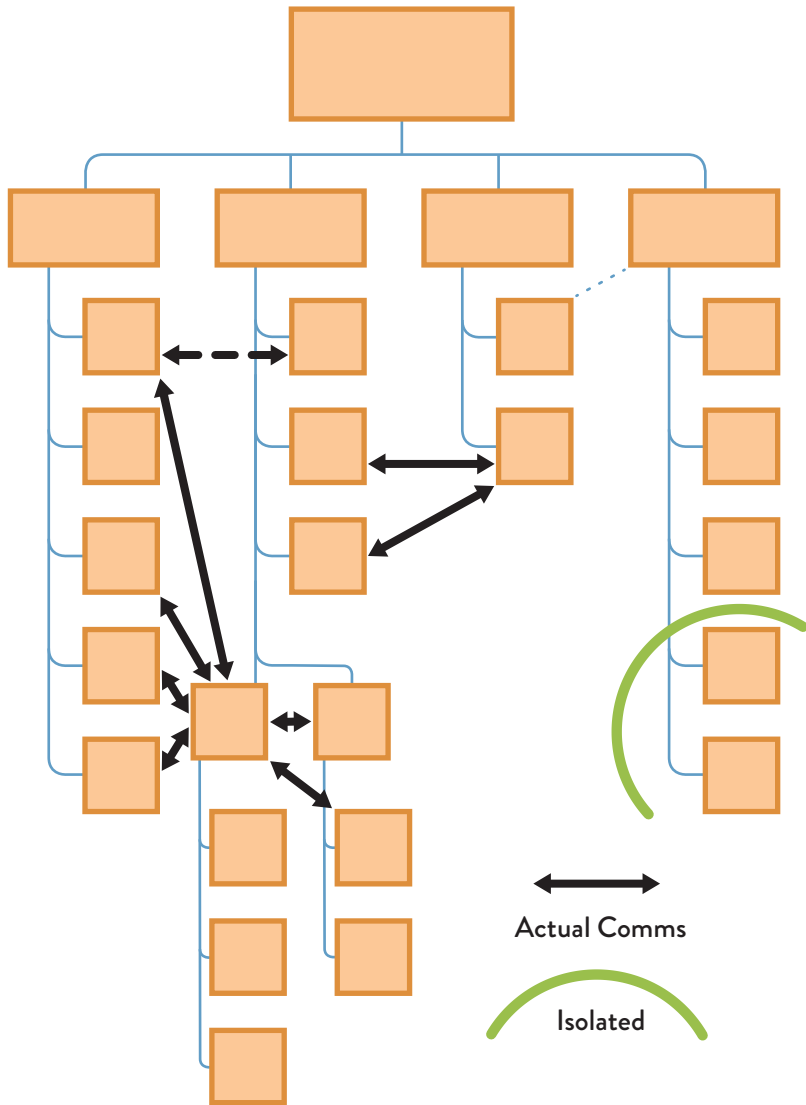


Figure 1.1: Org Chart with Actual Lines of Communication

In practice, people communicate laterally or “horizontally” with people from other reporting lines in order to get work done. This creativity and problem solving needs to be nurtured for the benefit of the organization, not restricted to optimize for top-down/bottom-up communication and reporting.

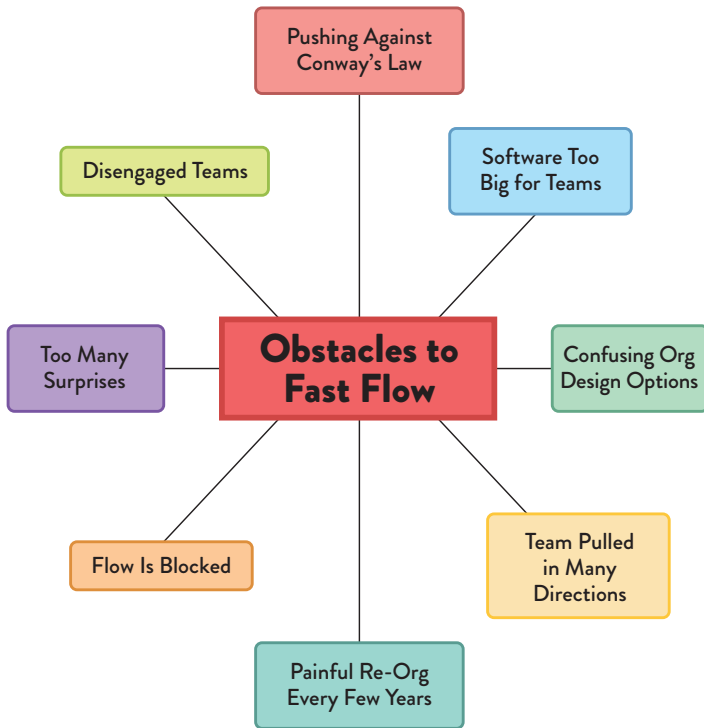


Figure 1.2: Obstacles to Fast Flow

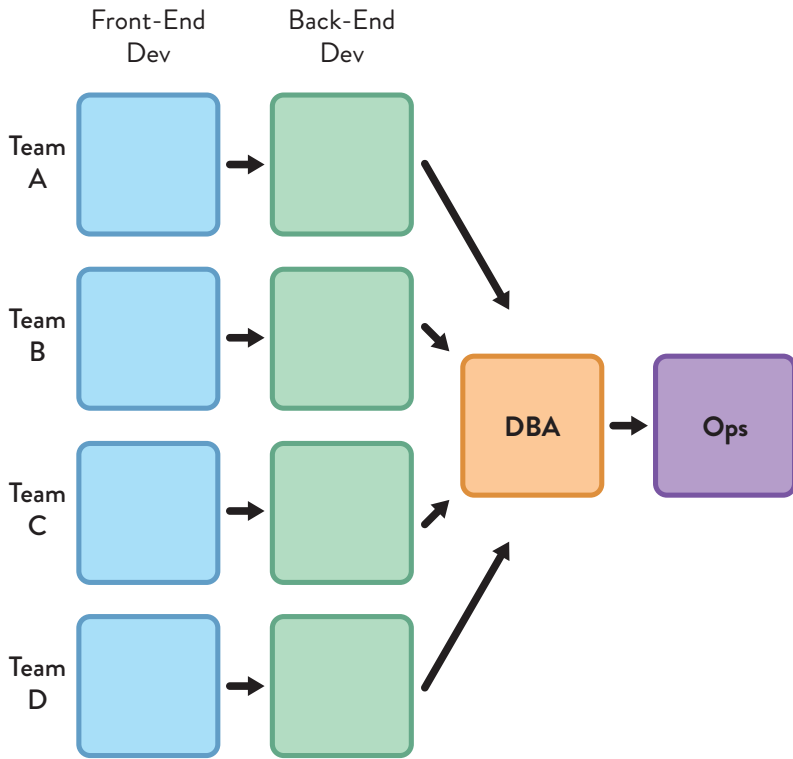


Figure 2.1: Four Teams Working on a Software System

Four separate teams consisting of front-end and back-end developers work on a software system. Front-end devs communicate only with back-end devs, who communicate with a single DBA for the database changes.

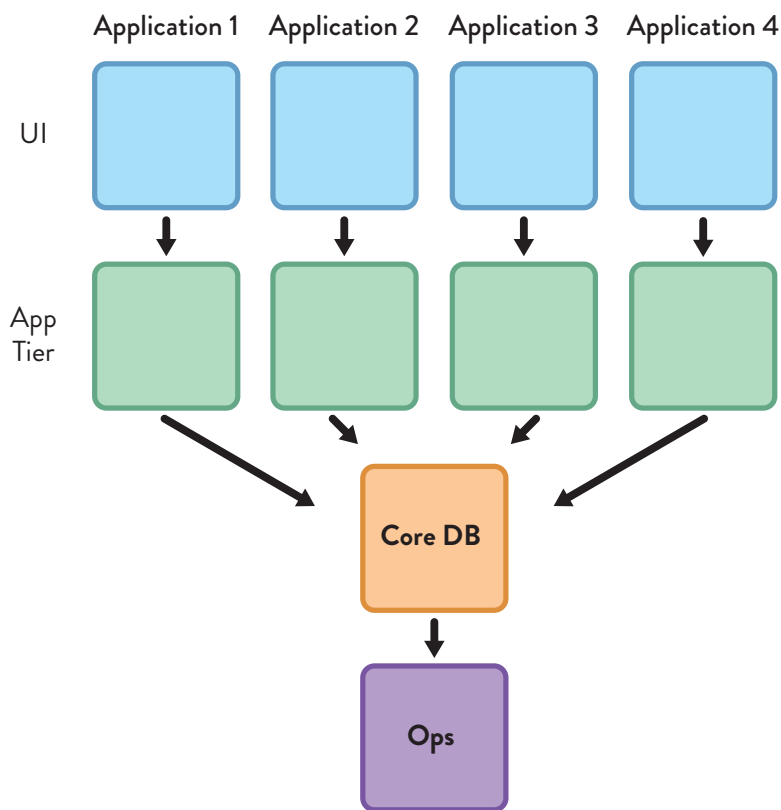


Figure 2.2: Software Architecture from Four-Team Organization

Four separate applications, each with a separate user interface (UI) and a back-end application tier that communicate with a single shared database. This reflects and matches the team communication architecture from Figure 2.1; the diagram has simply been rotated ninety degrees.

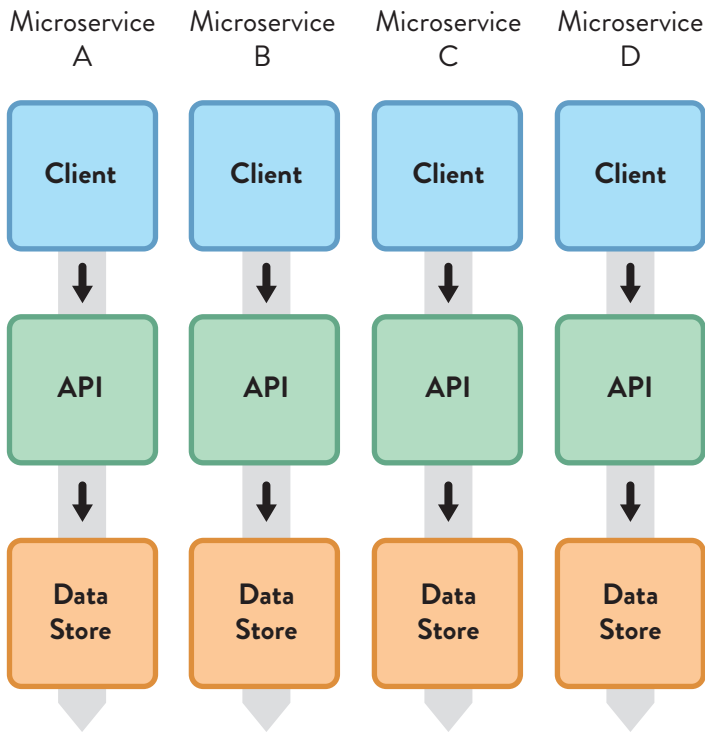


Figure 2.3: Microservices Architecture with Independent Services and Data Stores

A microservices-based architecture with four separate services, each with its own data store, API layer, and front-end client.

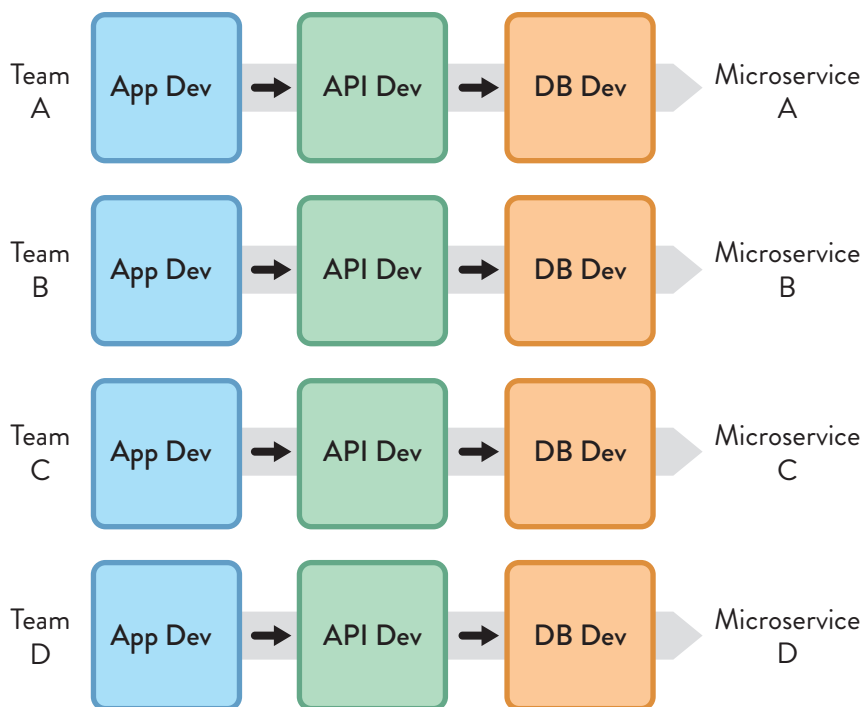


Figure 2.4: Team Design for Microservices Architecture with Independent Services and Data Stores

An organization design that anticipates the homomorphic force behind Conway's law to help produce a software architecture with four independent microservices. (Again, this is basically the diagram in Figure 2.3 rotated ninety degrees.)

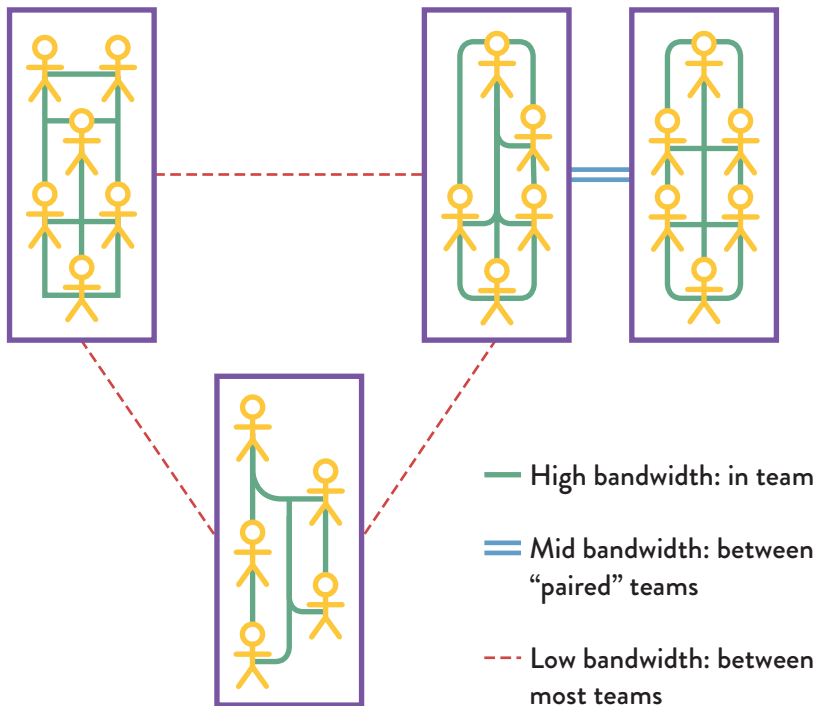


Figure 2.5: Inter-Team Communication

Communication within teams is high bandwidth. Communication between two “paired” teams can be mid bandwidth. Communication between most teams should be low bandwidth.

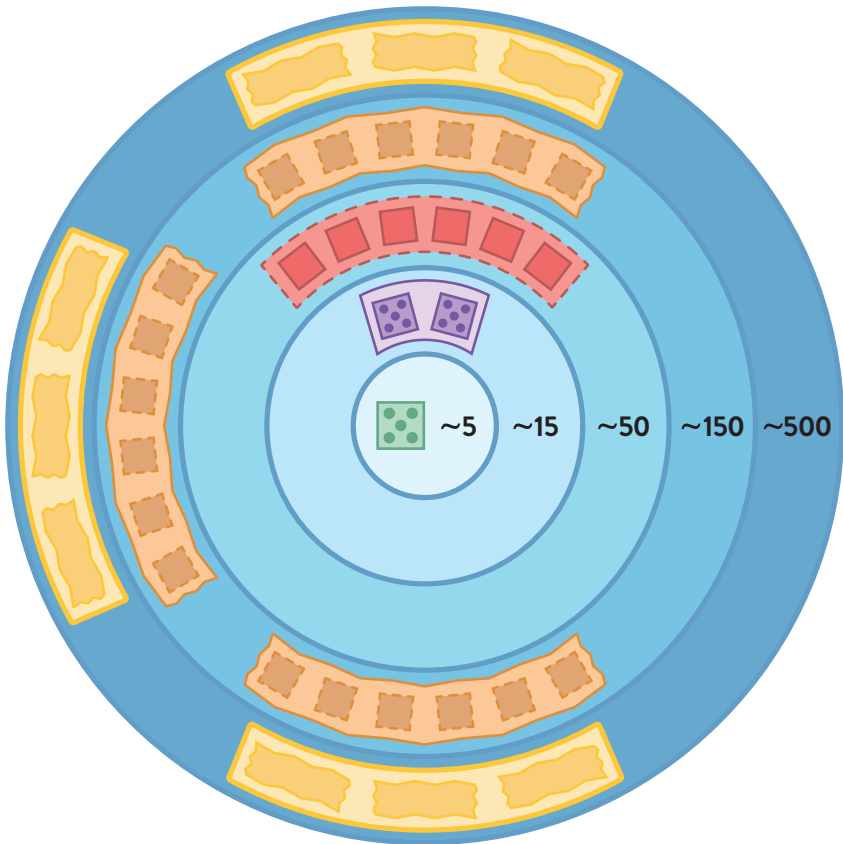
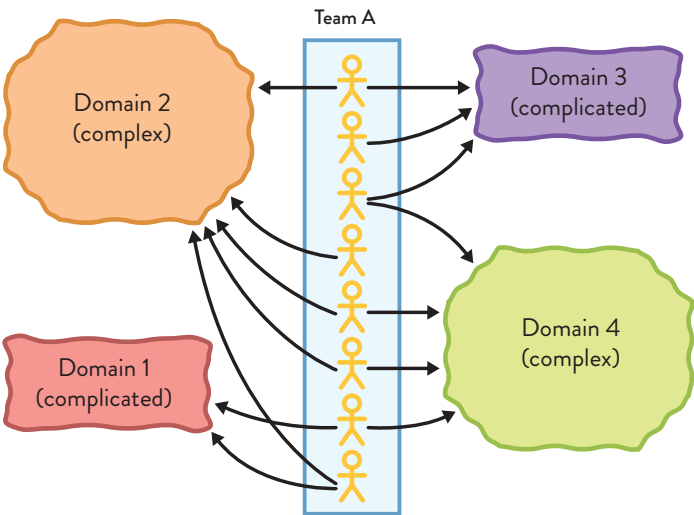


Figure 3.1: Scaling Teams Using Dunbar's Number

Organizational groupings should follow Dunbar's number, beginning with around five people (or eight for software teams), then increasing to around fifteen people, then fifty, then 150, then 500, and so on.

BEFORE



AFTER

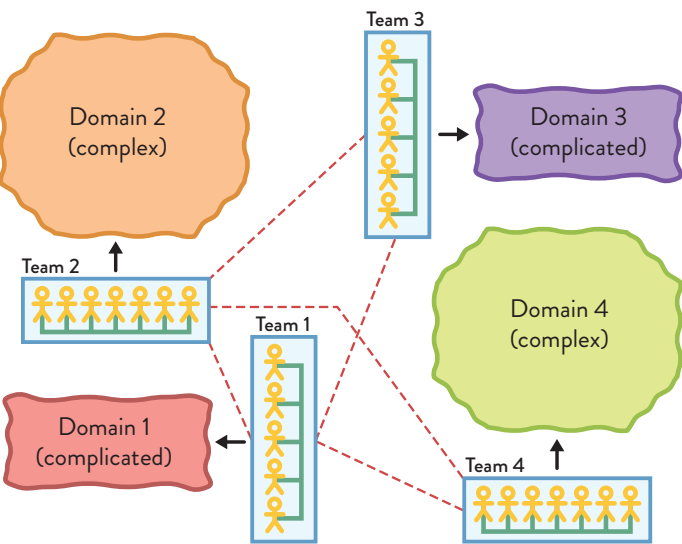


Figure 3.2: No More than One Complicated or Complex Domain per Team

Before: a larger team is spread thin across four domains (two complicated and two complex) and struggles to perform well. Intra-team morale is negatively affected, with frequent context switches and individual disengagement. After: with multiple smaller teams each focusing on a single domain, motivation rises and the team delivers faster and more predictably.

Low bandwidth inter-team collaboration allows solving occasional issues affecting two or more domains.

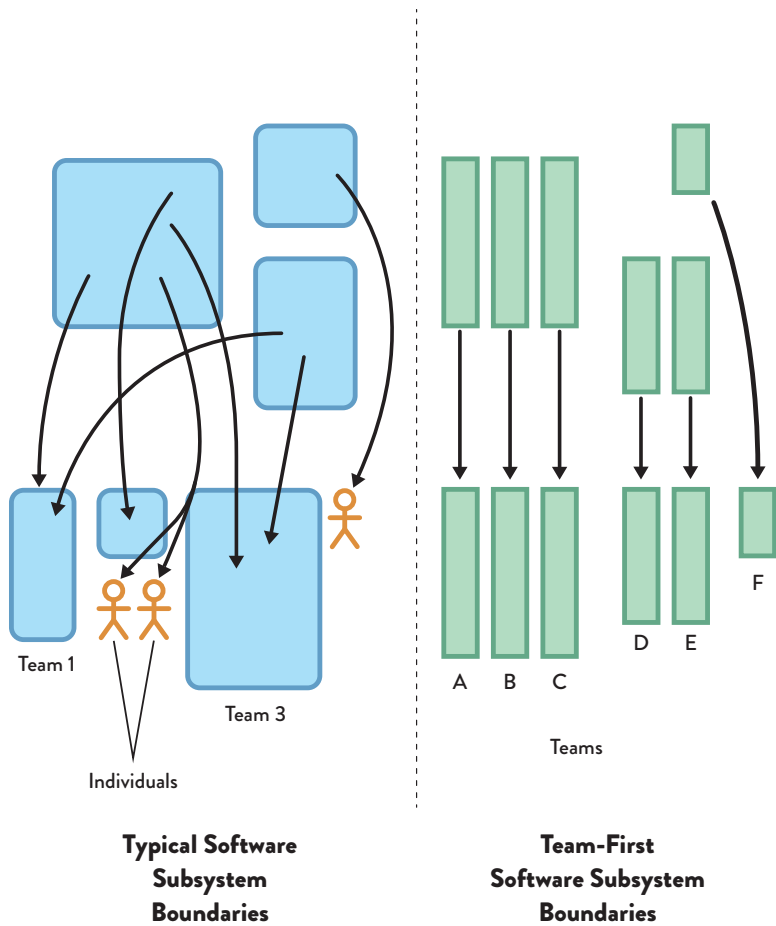


Figure 3.3: Typical vs. Team-First Software Subsystem Boundaries

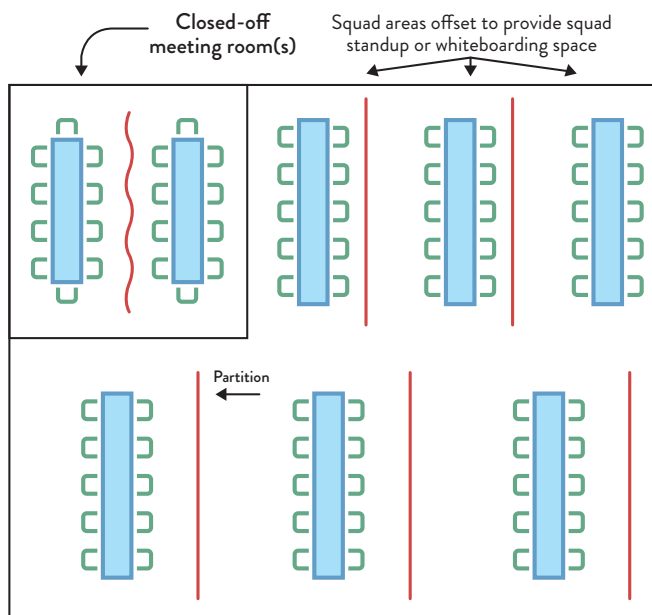


Figure 3.4: Office Layout at CDL

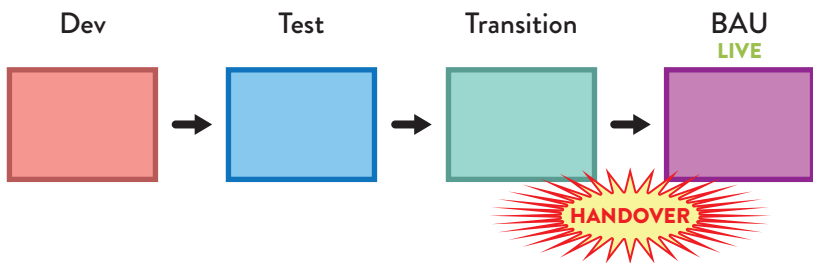


Figure 4.1: Organization not Optimized for Flow of Change

Traditional flow of change in an organization not optimized for flow, with a series of groups owning different activities and handing over the work to the next team.

No information flows back from the live systems into teams building the software.

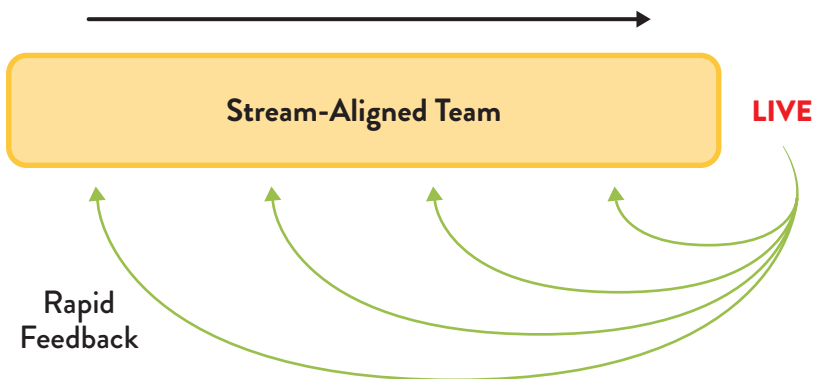


Figure 4.2: Organization Optimized for Flow of Change

Organizations set up for fast flow avoid hand-offs by keeping work within the stream-aligned team, and they ensure that the rich set of operational information flows back into the team.

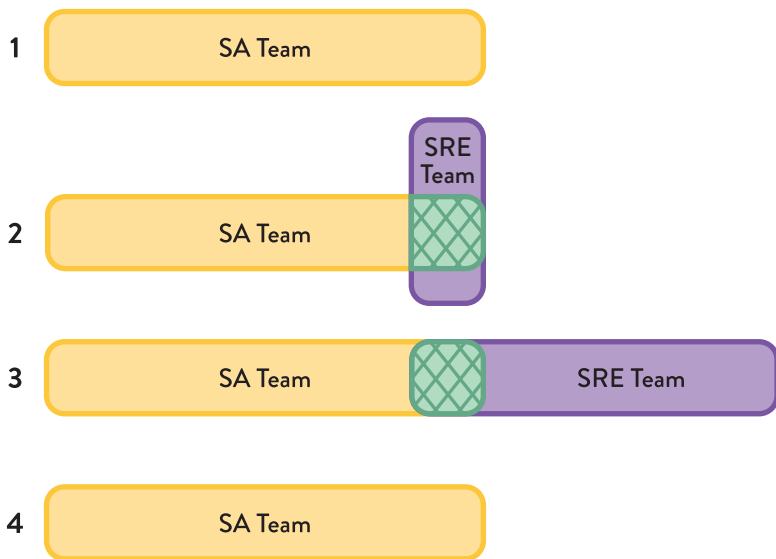


Figure 4.3: Relationship between SRE Team and Application Team

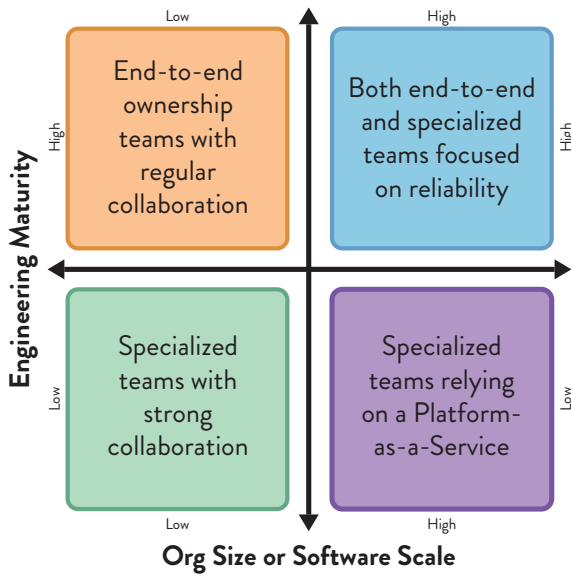


Figure 4.4: Influence of Size and Engineering Maturity on Choice of Topologies

Organization size (or software scale) and engineering discipline influence the effectiveness of team interaction patterns.

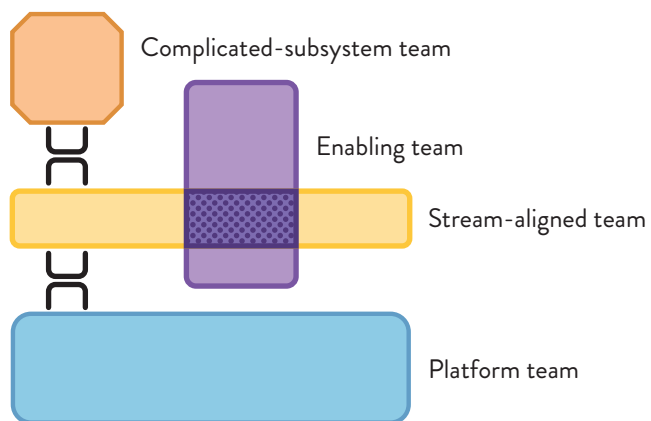


Figure 5.1: The Four Fundamental Team Topologies

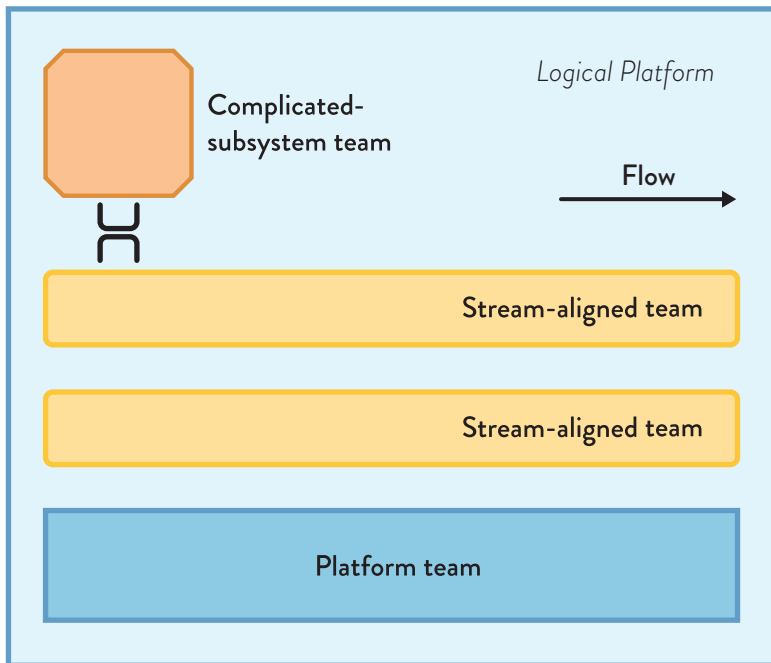


Figure 5.2: Platform Composed of Several Fundamental Team Topologies

In a large organization, the platform is composed of several other fundamental team topologies: stream-aligned Dev teams, complicated-subsystem teams, and a lower-level platform.

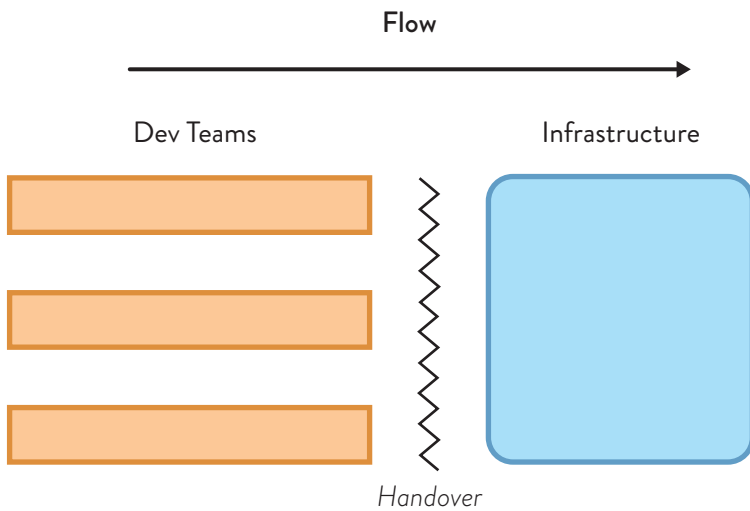


Figure 5.3: Traditional Infrastructure Team Organization

Many traditional infrastructure teams (on the right) blocked flow by being responsible for all changes to production infrastructure, including application changes, frequently gated by ITIL change processes. Work from Dev teams (on the left) was handed over to infrastructure or Ops teams for deployment, preventing flow.

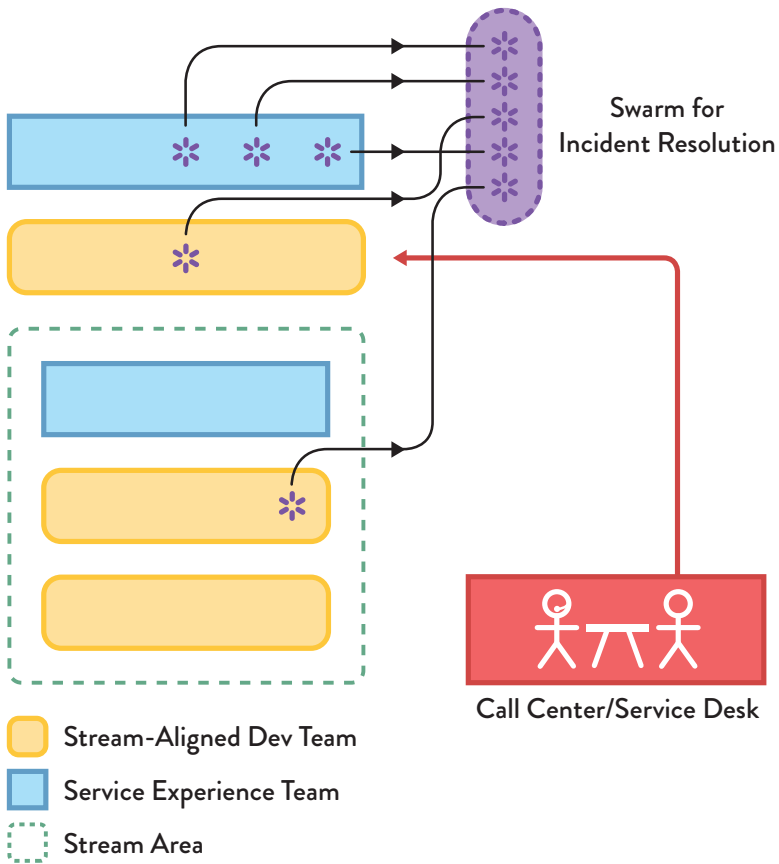


Figure 5.4: Support Teams Aligned to Stream of Change

The new model for support teams: aligned to the flow of change, usually paired with one or more stream-aligned Dev teams. Incidents are handled with dynamic “swarming.”

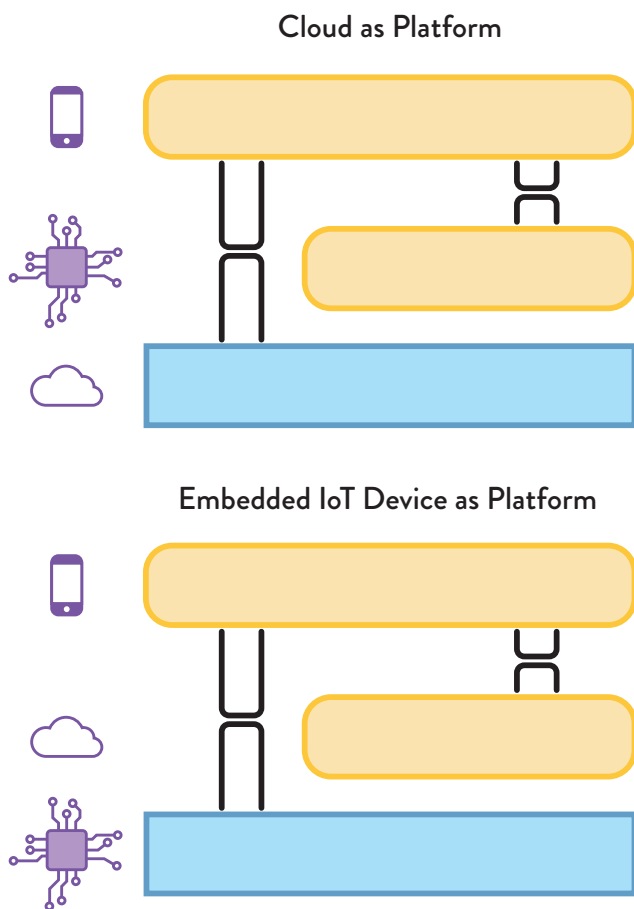


Figure 6.1: Mobile, Cloud, and IoT Technology Fracture Plane Scenario
With three very disparate technologies (mobile, cloud, and IoT), an organization must decide on an approach to fracture planes that makes sense based on the cognitive load and the change cadence in each area.

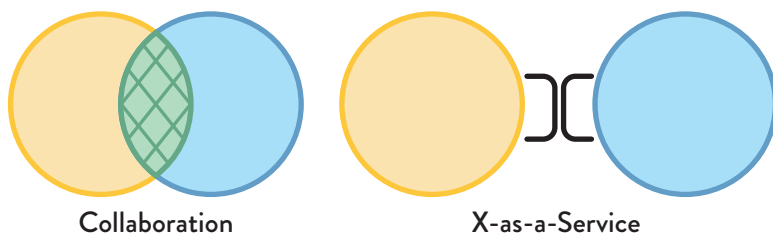


Figure 7.1: Collaboration vs. X-as-a-Service

Collaboration means explicitly working together on defined areas. X-as-a-Service means one team consumes something “as a service” from another team.



Figure 7.2: The Three Team Interaction Modes

Collaboration mode is shown with diagonal cross-hatching, X-as-a-Service mode is shown with brackets, and facilitating is shown with dots.

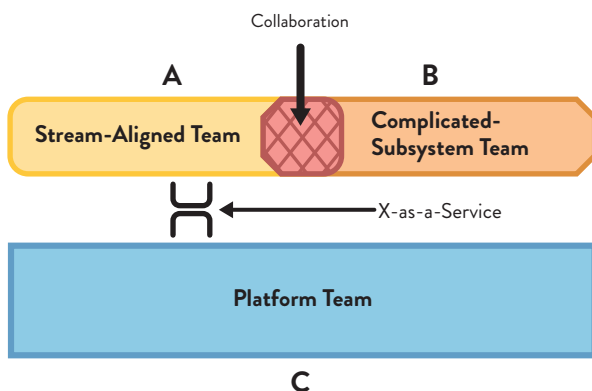


Figure 7.3: Team Interaction Modes Scenario

Stream-aligned Team A collaborates with complicated-subsystem Team B (shown with cross-hatching) while also consuming the platform provided by Team C, using the X-as-a-Service mode (shown with brackets).

Table 7.1: Advantages and Disadvantages of Collaboration Mode

Advantages	Disadvantages
• Rapid innovation and discovery • Fewer hand-offs	• Wide, shared responsibility for each team • More detail/context needed between teams, leading to higher cognitive load • Possible reduced output during collaboration compared to before
Constraint: A team should use collaboration mode with, at most, one other team at a time. A team should not use collaboration with more than one team at the same time.	
Typical Uses: Stream-aligned teams working with complicated-subsystem teams; stream-aligned teams working with platform teams; complicated-subsystem teams working with platform teams	

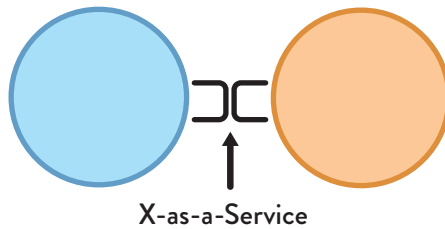


Figure 7.4: X-as-a-Service Team Interaction Mode

In this case, the team on the right is providing something “as a service” to the team on the left (perhaps an API, some developer tooling, or even an entire platform).

Table 7.2: Advantages and Disadvantages of X-as-a-Service Mode

Advantages	Disadvantages
• Clarity of ownership with clear responsibility boundaries • Reduced detail/context needed between teams, so cognitive load is limited	• Slower innovation of the boundary or API • Danger of reduced flow if the boundary or API is not effective
Constraint: A team should expect to use the X-as-a-Service interaction with many other teams simultaneously, whether consuming or providing a service.	
Typical Uses: Stream-aligned teams and complicated-subsystem teams consuming Platform-as-a-Service from a platform team; stream-aligned teams and complicated-subsystem teams consuming a component or library as a service from a complicated-subsystem team.	

Table 7.3: Advantages and Disadvantages of Facilitation Mode

Advantages	Disadvantages
• Unblocking of stream-aligned teams to increase flow • Detection of gaps and misaligned capabilities or features in components and platforms	• Requires experienced staff to not work on “building” or “running” things • The interaction may be unfamiliar or strange to one or both teams involved in facilitation
Constraint: A team should expect to use the facilitating interaction mode with a small number of other teams simultaneously, whether consuming or providing the facilitation.	
Typical Uses: An enabling team helping a stream-aligned, complicated-subsystem, or platform team; or a stream-aligned, complicated-subsystem, or platform team helping a stream-aligned team.	

Table 7.4: Team interaction modes of the fundamental team topologies

	Collaboration	X-as-a-Service	Facilitating
Stream-aligned	Typical	Typical	Occasional
Enabling	Occasional		Typical
Complicated-subsystem	Occasional	Typical	
Platform	Occasional	Typical	

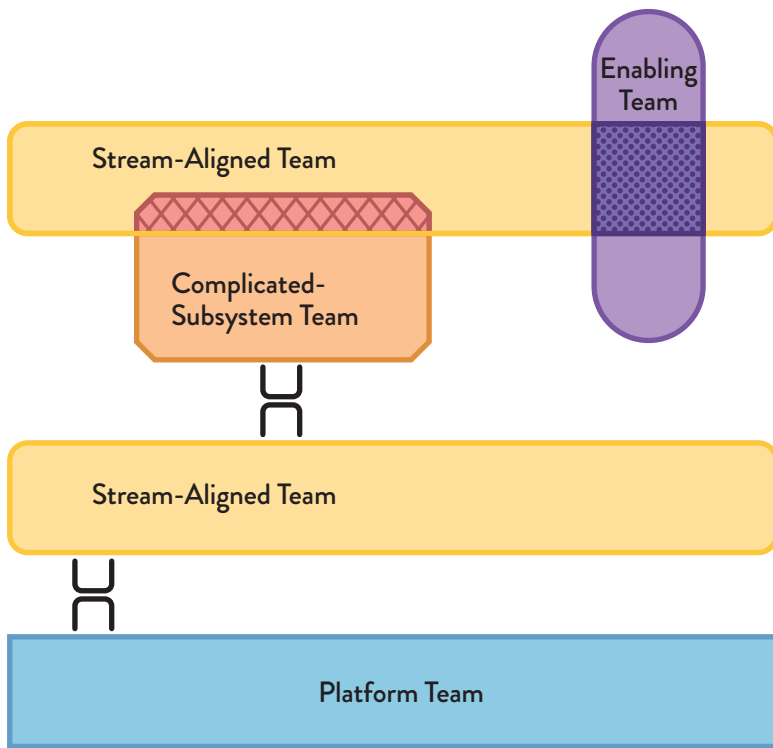


Figure 7.5: Primary Interaction Modes for the Four Fundamental Team Topologies

Stream-aligned teams use X-as-a-Service or collaboration; enabling teams use facilitation; complicated-subsystem teams use X-as-a-Service; platform teams use X-as-a-Service for teams that consume the platform.

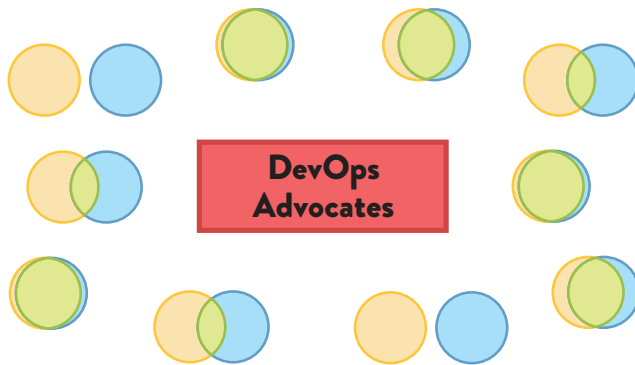


Figure 7.6: Team Interaction Modes at IBM around 2014

Team interaction modes at IBM around 2014, with a team of “DevOps advocates” coordinating and facilitating learning and team changes.

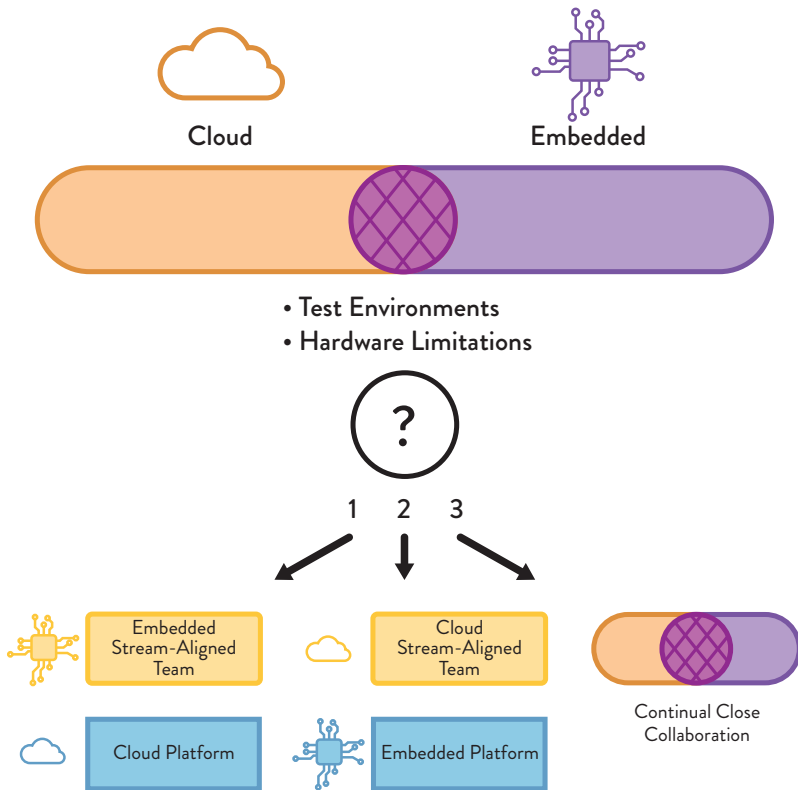


Figure 8.1: Collaboration between Cloud and Embedded Teams

Two teams (“cloud” and “embedded”) collaborate to share practices and increase awareness. The results will include heightened awareness of the options for future team interactions: (1) treat the cloud software as a platform for the embedded team to use, (2) treat the embedded devices as a platform for the cloud team to use, or (3) continue with close collaboration.

Expected Evolution (2014)

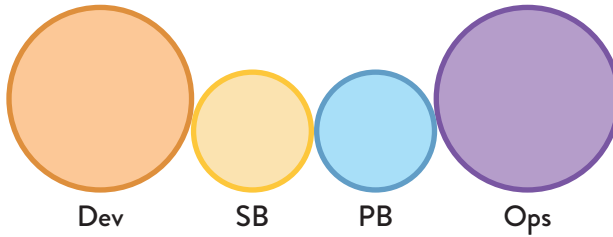


Figure 8.2: System-Build and Platform-Build Team at TransUnion

A team from Dev (SB) and a team from Ops (PB) exploring close interactions.

Expected in 6+ months; Actual realization 2 years

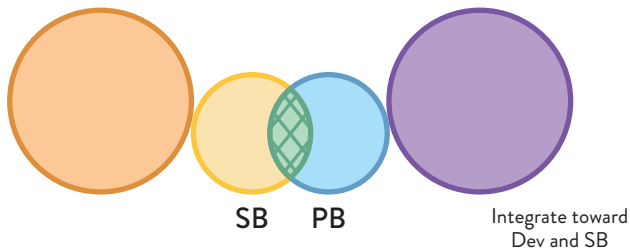


Figure 8.3: System-Build and Platform-Build Team Collaboration at TransUnion

The two teams, SB and PB, collaborating closely.

Expected in 12+ months; Actual realization 4 years

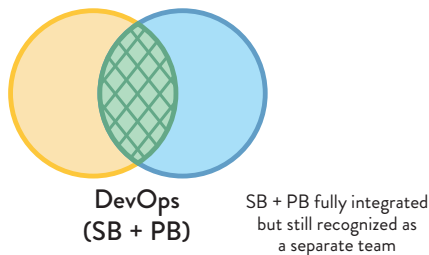


Figure 8.4: System-Build and Platform-Build Teams Merged at TransUnion

The SB and PB teams merged, helping to bring Dev and Ops together.

2018

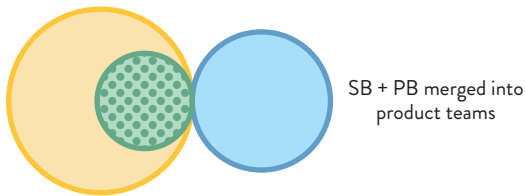


Figure 8.5: System-Build and Platform-Build Teams Merged Back into Dev and Ops at TransUnion

The SB and PB teams merged back into Dev and Ops, providing Platform-as-a-Service.



Figure 8.6: Evolution of Team Topologies

The evolution of Team Topologies from close collaboration to limited collaboration (discovery) through to X-as-a-Service for established, predictable delivery.

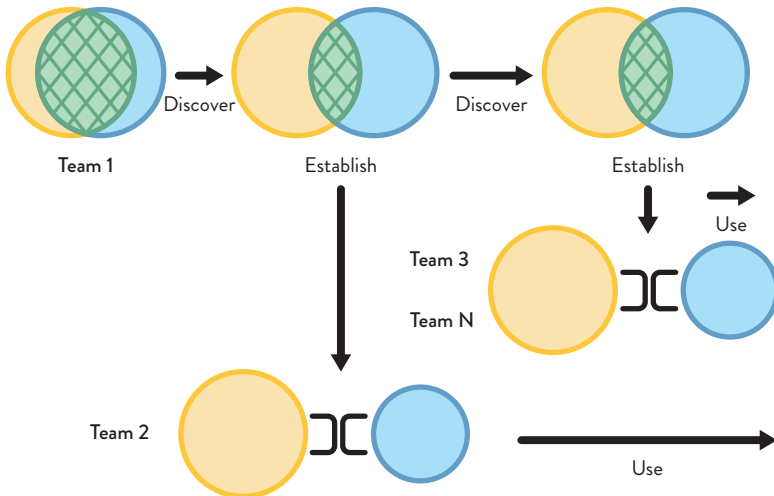


Figure 8.7: Evolution of Team Topologies in an Enterprise

Team 1 continues to collaborate with a platform team, discovering new patterns and ways of using new technologies. This discovery activity eventually enables Team 2 to adopt an X-as-a-Service relationship with the platform team. Later, Teams 3 and beyond adopt a later version of the platform, using it as a service without having to collaborate closely with the platform team.

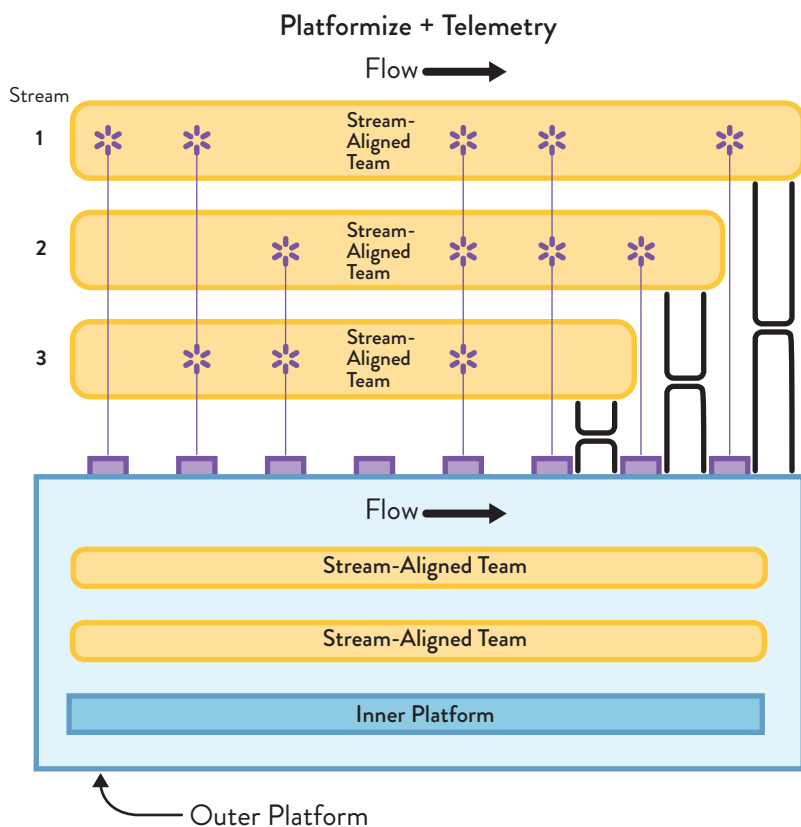


Figure 8.8: Example of a “Platform Wrapper”

Increase flow predictability in higher-level business services (streams) through the use of a “platform wrapper” to “platformize” the lower-level services and APIs, allowing the streams to treat all their dependencies as a single platform with a holistic roadmap and consistent DevEx. The streams also have rich telemetry to track flow and resource usage of the platform.

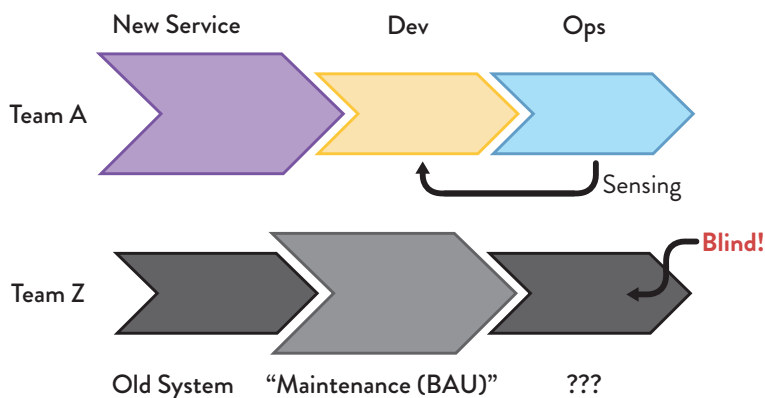


Figure 8.9: New-Service and “Business as Usual” (BAU) Teams

Having separate teams for “new stuff” and BAU tends to prevent learning, improvements and ability to self-steer. It is a non-cybernetic approach.

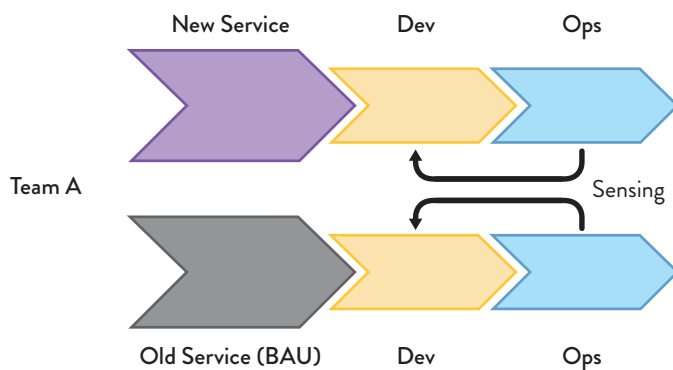


Figure 8.10: Side-by-Side New Service and BAU Teams

A cybernetic approach to maintaining older systems has a single stream-aligned team (or pair of teams) developing and running the new service *and* the older systems, enabling the team to retro-fit newer telemetry to the older system and increase the fidelity of the sensing from both systems.

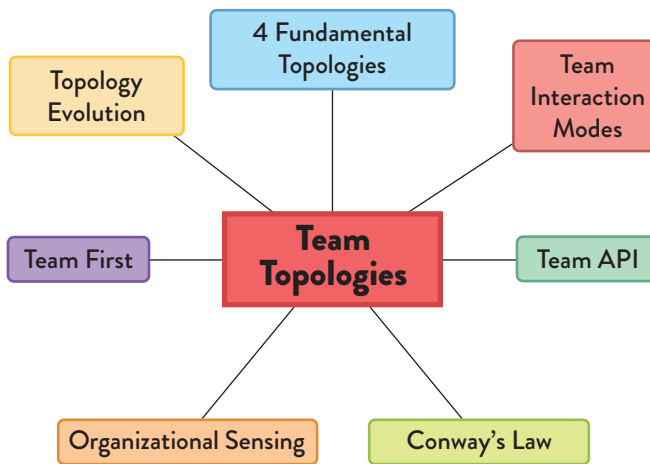


Figure 9.1: Core Ideas of Team Topologies

GLOSSARY

API (application programming interface): a description and specification for how to interact programmatically with software.

application monolith: a single, large application with many dependencies and responsibilities that possibly exposes many services and/or different user journeys.

bounded context: a unit for partitioning a larger domain (or system) model into smaller parts, each of which represents an internally consistent business domain area.

Brooks's law: law coined by Fred Brooks which states that adding new people to a team doesn't immediately increase the capacity of a team.

cognitive load: the amount of working memory being used.

collaboration mode: team(s) working closely together with another team.

complicated-subsystem team: responsible for building and maintaining a part of the system that depends heavily on specialist knowledge.

Conway's law: law coined by Mel Conway that states that system design will copy the communication structures of the organization which designs it.

domain complexity: how complex the problem is that is being solved via software.

Dunbar's number: coined by anthropologist Robin Dunbar, which states that fifteen is the limit of people one person can trust; of those, only around five can be known and trusted closely.

enabling team: team(s) composed of specialists in a given technical (or product) domain; they help bridge the capability gap.

extraneous cognitive load: relates to the environment in which the task is being done (e.g., "How do I deploy this component, again?" "How do I configure this service?").

facilitating mode: team(s) helping (or being helped by) another team to clear impediments.

flow of change: a stream of related updates or alterations to a software service or system, usually aligned to user goals or other core focus of the business.

fracture plane: a natural “seam” in the software system that allows it to be easily split into two or more parts.

germane cognitive load: relates to aspects of the task that need special attention for learning or high performance (e.g., “How should this service interact with the ABC service?”).

intrinsic cognitive load: relates to aspects of the task fundamental to the problem space (e.g., “What is the structure of a Java class?” “How do I create a new method?”).

joined-at-the-database monolith: composed of several applications or services all coupled to the same database schema, making them difficult to change, test, and deploy separately.

monolithic build: uses one gigantic continuous integration (CI) build to get a new version of a component.

monolithic model: software that attempts to force a single domain language and representation (format) across many different contexts.

monolithic release: a set of smaller components bundled together into a “release.”

monolithic thinking: “one-size-fits-all” thinking for teams that leads to unnecessary restrictions on technology and implementation approaches between teams.

monolithic workplace: a single office layout pattern for all teams and individuals in the same geographic location.

organizational sensing: teams and their internal and external communication are the “senses” of the organization (sight, sound, touch, smell, taste).

platform team: enables stream-aligned teams to deliver work with substantial autonomy.

reverse Conway maneuver: organizations should evolve their team and organizational structure to achieve the desired architecture.

stream-aligned team: a team aligned to a single, valuable stream of work.

team API: an API surrounding each team.

Team Topologies: model for organizational design that provides a key technology-agnostic mechanism for modern software-intensive enterprises to sense when a change in strategy is required (either from a business or technology point of view).

thinnest viable platform: a careful balance between keeping the platform small and ensuring that the platform is helping to accelerate and simplify software delivery for teams building on the platform.

X-as-a-Service mode: consuming or providing something with minimal collaboration.

RECOMMENDED READING

Key Management Concepts and Practices for Reliable, Fast Flow

- *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations* by Nicole Forsgren, PhD, Jez Humble, and Gene Kim (Portland, Oregon: IT Revolution, 2018).
- *Designing Delivery: Rethinking IT in the Digital Service Economy* by Jeff Sussna (Beijing: O'Reilly Media, 2015).
- *Fearless Change: Patterns for Introducing New Ideas* by Mary Lynn Manns and Linda Rising (Boston: Addison Wesley, 2004).

Key Practices and Approaches for Organizations, Software, and Systems

- *Team Genius: The New Science of High-Performing Organizations* by Rich Karlgaard and Michael S. Malone (New York, NY: HarperBusiness, 2015).
- *Agile Development in the Large: Diving into the Deep* by Jutta Eckstein (New York: Dorset House Publishing Co Inc.,US, 2004).
- *Domain-Driven Design: Tackling Complexity in the Heart of Software* by Eric Evans (Boston: Addison-Wesley, 2003).
- *Thinking in Promises* by Mark Burgess (Sebastopol, California: O'Reilly Media, 2015).

Key Engineering Practices that Enable Fast Flow

- *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation* by Jez Humble and David Farley (Upper Saddle River, NJ: Addison Wesley, 2010).

- *Release It! Design and Deploy Production-Ready Software* by Michael T. Nygard (Raleigh, North Carolina: O'Reilly, 2018).
- *Team Guide to Software Operability*, Team Guide Series 1, by Matthew Skelton and Rob Thatcher (Leeds, UK: Conflux Books, 2016).
- *Team Guide to Software Testability*, Team Guide Series 3, by Ash Winter and Rob Meaney (Leeds, UK: Conflux Books, 2018).
- *Team Guide to Software Releasability*, Team Guide Series 4, by Manuel Pais and Chris O'Dell (Leeds, UK: Conflux Books, 2018).

REFERENCES

- Ackoff, Russell L. *Re-Creating the Corporation: A Design of Organizations for the 21st Century*. Oxford: Oxford University Press, 1999.
- Ackoff, Russell L., Herbert J. Addison, and Sally Bibb. *Management F-Laws: How Organizations Really Work*. United Kindgom, Triarchy Press, 2007.
- Adams, Paul. "Scaling Product Teams: How to Build and Structure for Hypergrowth." *Inside Intercom* (blog). January 28, 2015. <https://www.intercom.com/blog/how-we-build-software/>.
- Adkins, Lyssa. *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Upper Saddle River, NJ: Addison-Wesley Professional, 2010.
- Allen, Thomas J. *Managing the Flow of Technology*. Cambridge, MA: MIT Press, 1984.
- Allspaw, John. "Blameless PostMortems and a Just Culture." *Code as Craft* (blog), May 22, 2012. <https://codeascraft.com/2012/05/22/blameless-postmortems/>.
- Almeida, Thiago. "DevOps Lessons Learned at Microsoft Engineering." *InfoQ*, May 22, 2016. <https://www.infoq.com/articles/devops-lessons-microsoft>.
- Ancona, Deborah Gladstein, and David F. Caldwell. "Demography and Design: Predictors of New Product Team Performance." *Organization Science* 3 no. 3 (1992): 321–341. <https://doi.org/10.1287/orsc.3.3.321>.
- Axelrod, Robert A. *Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton, NJ: Princeton University Press, 1997.
- Bauernberger, Joachim. "DevOps in Telecoms—Is It Possible?" *Telecom Tech News*, October 1, 2014. <http://www.telecomstechnews.com/news/2014/oct/01/devops-telecoms-it-possible/>.
- Beal, Helen. "The Industry Just Can't Decide about DevOps Teams." *InfoQ*, October 26, 2017. <https://www.infoq.com/news/2017/10/devops-teams-good-or-bad>.
- Beer, Stafford. *Brain of the Firm*, 2nd edition. Chichester, UK: John Wiley & Sons, 1995.
- Bennett, Drake. "The Dunbar Number, From the Guru of Social Networks." *Bloomberg.com*, January 11, 2013. <http://www.bloomberg.com/news/articles/2013-01-10/the-dunbar-number-from-the-guru-of-social-networks>.
- Bernstein, Ethan, John Bunch, Niko Canner, and Michael Lee. "Beyond the Holacracy Hype." *Harvard Business Review*, July 1, 2016. <https://hbr.org/2016/07/beyond-the-holacracy-hype>.
- Bernstein, Ethan, Jesse Shore, and David Lazer. "How Intermittent Breaks in Interaction Improve Collective Intelligence." *Proceedings of the National Academy of Sciences* 115 no. 35 (August, 2018): 8734–8739. <https://doi.org/10.1073/pnas.1802407115>.

- Bernstein, Ethan S., and Stephen Turban. "The Impact of the 'Open' Workspace on Human Collaboration." *Philosophical Transactions of the Royal Society B* 373 no. 1753 (2018). <https://doi.org/10.1098/rstb.2017.0239>.
- Betz, Charles. *Managing Digital: Concepts and Practices*. The Open Group, 2018.
- Beyer, Betsy, Jennifer Petoff, Chris Jones, and Niall Richard Murphy (eds). *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly, 2016.
- Blalock, Micah. "Of Mustard Seeds and Microservices." *Credera* (blog), May 6, 2015. <https://www.credera.com/blog/technology-insights/java/mustard-seeds-microservices/>.
- Bosch, Jan. "On the Development of Software Product-Family Components." In *Software Product Lines*, edited by Robert L. Nord, 146–164. Berlin: Springer, 2004.
- Bottcher, Evan. "What I Talk About When I Talk About Platforms." *MartinFowler.com* (blog), March 5, 2018. <https://martinfowler.com/articles/talk-about-platforms.html>.
- Brandolini, Alberto. "Strategic Domain Driven Design with Context Mapping." *InfoQ*, November 25, 2009. <https://www.infoq.com/articles/ddd-contextmapping>.
- Bright, Peter. "How Microsoft Dragged Its Development Practices into the 21st Century." *Ars Technica*, August 6, 2014. <https://arstechnica.com/information-technology/2014/08/how-microsoft-dragged-its-development-practices-into-the-21st-century/>.
- Brooks, Fred. *The Mythical Man-Month: Essays on Software Engineering*. Boston, MA: Addison-Wesley, 1995.
- Brown, Simon. "Are You a Software Architect?" *InfoQ*, February 9, 2010. <https://www.infoq.com/articles/brown-are-you-a-software-architect>.
- Bryson, Brandon. "Architects Should Code: The Architect's Misconception." *InfoQ*, August 6, 2015. <https://www.infoq.com/articles/architects-should-code-bryson>.
- Burgess, Mark. *Thinking in Promises: Designing Systems for Cooperation*. Sebastopol, CA: O'Reilly Media, 2015.
- Carayon, Pascale. "Human Factors of Complex Sociotechnical Systems." *Applied Ergonomics, Special Issue: Meeting Diversity in Ergonomics* 37 no. 4 (2006): 525–535. <https://doi.org/10.1016/j.apergo.2006.04.011>.
- Casella, Karen. "Improving Team Productivity by Reducing Context Switching | LinkedIn." *LinkedIn Pulse*, October 26, 2016. <https://www.linkedin.com/pulse/improving-team-productivity-reducing-context-karen-casella/>.
- Chaudhary, Mukesh. "Working with Component Teams: How to Navigate the Complexity-Scrum Alliance." *ScrumAlliance.org*, September 5, 2012. <https://www.scrumalliance.org/community/member-articles/301>.
- Cherns, Albert. "The Principles of Sociotechnical Design." *Human Relations* 29 no. 8 (1976): 783–792. <https://doi.org/10.1177/001872677602900806>.
- Clegg, Chris W. "Sociotechnical Principles for System Design." *Applied Ergonomics* 31 no. 5 (2000): 463–477. [https://doi.org/10.1016/S0003-6870\(00\)00009-0](https://doi.org/10.1016/S0003-6870(00)00009-0).
- Cockcroft, Adrian. "Goto Berlin—Migrating to Microservices (Fast Delivery)." Presented at the GOTO Berlin conference, Berlin, November 15, 2014. <http://www.slideshare.net/adriancockcroft/goto-berlin>.
- Cohn, Mike. "Nine Questions To Assess Scrum Team Structure." *Mountain Goat Software* (blog), March 9, 2010. <https://www.mountaingoatsoftware.com/blog/nine-questions-to-assess-team-structure>.

- Conway, Melvin E. "How Do Committees Invent? Design Organization Criteria." *Datamation*, 1968.
- Conway, Mel. "Toward Simplifying Application Development, in a Dozen Lessons," MelConway.com, January 3, 2017. <http://melconway.com/Home/pdf/simplify.pdf>.
- Cooley, Faith. "Organizational Design for Effective Software Development." SlideShare, posted by Dev9Com, November 12, 2014. <http://www.slideshare.net/Dev9Com/organizational-design-for-effective-software-development>.
- Coplien, James O., and Neil Harrison. *Organizational Patterns of Agile Software Development*. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.
- Cottmeyer, Mike. "Things to Consider When Structuring Your Agile Enterprise." *LeadingAgile* (blog), February 5, 2014. <https://www.leadingagile.com/2014/02/structure-agile-enterprise/>.
- Coutu, Diane. "Why Teams Don't Work." *Harvard Business Review*, May 1, 2009. <https://hbr.org/2009/05/why-teams-dont-work>.
- Crawford, Jason. "Amazon's 'Two-Pizza Teams': The Ultimate Divisional Organization." *JasonCrawford.org* (blog), July 30, 2013. <http://blog.jasoncrawford.org/two-pizza-teams>.
- Cunningham, Ward. "Understand the High Cost of Technical Debt by Ward Cunningham—DZone Agile." *Dzone.com*, August 24, 2013. <https://dzone.com/articles/understand-high-cost-technical>.
- Cusumano, Michael A. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People*, 1st Touchstone edition. New York: Simon and Schuster, 1988.
- Cutler, John. "12 Signs You're Working in a Feature Factory." *Hacker Noon*, November 17, 2016. <https://hackernoon.com/12-signs-youre-working-in-a-feature-factory-44a5b938d6a2>.
- Davies, Rachel, and Liz Sedley. *Agile Coaching*. Raleigh, NC: Pragmatic Bookshelf, 2009.
- DeGrandis, Dominica. *Making Work Visible: Exposing Time Theft to Optimize Workflow*. Portland, OR: IT Revolution Press, 2017.
- DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams*, 2nd revised edition. New York, NY: Dorset, 1999.
- Deming, W. Edwards. *Out of the Crisis*. Cambridge, MA: MIT Press, 1986.
- DeSanctis, Gerardine, and Marshall Scott Poole. "Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory." *Organization Science* 5 no. 2 (May 1994): 121–147.
- Dogan, Jaana B. "The SRE Model." *Medium*, July 31, 2017. <https://medium.com/@rakyll/the-sre-model-6e19376ef986>.
- Doorley, Scott, and Scott Witthoft. *Make Space: How to Set the Stage for Create Collaboration*. Hoboken, NJ: John Wiley & Sons, 2012.
- Driskell, James E., and Eduardo Salas. "Collective Behavior and Team Performance." *Human Factors* 34 no. 3 (1992): 277–288. <https://doi.org/10.1177/001872089203400303>.
- Driskell, James E., Eduardo Salas, and Joan Johnston. "Does Stress Lead to a Loss of Team Perspective?" *Group Dynamics: Theory, Research, and Practice* 3, no. 4 (1999): 291–302.
- Drucker, Peter. *The Daily Drucker: 366 Days of Insight and Motivation for Getting the Right Things Done*. New York: HarperCollins, 2018.
- Dunbar, R. I. M. "Neocortex Size as a Constraint on Group Size in Primates." *Journal of Human Evolution* 22, no. 6 (1992): 469–493. [https://doi.org/10.1016/0047-2484\(92\)90081-J](https://doi.org/10.1016/0047-2484(92)90081-J).

- Dunbar, Professor Robin. *How Many Friends Does One Person Need?: Dunbar's Number and Other Evolutionary Quirks*. London: Faber & Faber, 2010.
- Eckstein, Jutta. *Agile Development in the Large: Diving into the Deep*. New York: Dorset, 2004.
- Eckstein, Jutta. "Architecture in Large Scale Agile Development." In *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, edited by Torgeir Dingsøy, Nils Brede Moe, Roberto Tonelli, Steve Counsell, Cigdem Gencel, and Kai Petersen. Switzerland, Springer International Publishing, 2014.
- Edmondson, Amy. "Psychological Safety and Learning Behavior in Work Teams." *Administrative Science Quarterly* 44 no. 2 (1999): 350–383. <https://doi.org/10.2307/2666999>.
- Edmondson, Amy C. *Managing the Risk of Learning: Psychological Safety in Work Teams*. In *International Handbook of Organization Teamwork and Cooperative Working*, edited by Michael A. West, Dean Tjosvold, and Ken G. Smith. Hoboken, NJ: Wiley & Sons, 2003.
- Edwards, Damon. "What is DevOps?" Dev2Ops.org, February 23, 2010. <http://dev2ops.org/g/2010/02/what-is-devops>.
- The Essential Elements of Enterprise PaaS*. Palo Alto, CA: Pivotal, 2015. <https://content.pivotal.io/white-papers/the-essential-elements-of-enterprise-paas>.
- Evans, Eric. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison Wesley, 2003.
- Evans, William. "The Need for Speed: Enabling DevOps through Enterprise Architecture | #DOES16." SlideShare, posted by William Evans, November 2, 2016. <https://www.slideshare.net/willewans/the-need-for-speed-enabling-devops-through-enterprise-architecture>.
- Fan, Xiaocong, Po-Chun Chen, and John Yen. "Learning HMM-Based Cognitive Load Models for Supporting Human-Agent Teamwork." *Cognitive Systems Research* 11, no. 1 (2010): 108–119.
- Feathers, Michael. *Working Effectively with Legacy Code*. Upper Saddle River, NJ: Prentice Hall, 2004.
- Forrester, Russ, and Allan B. Drexler. "A Model for Team-Based Organization Performance." *The Academy of Management Executive* 13 no. 3 (1999), 36–49.
- Forsgren, PhD, Nicole, Jez Humble, and Gene Kim. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. Portland, Oregon: IT Revolution Press, 2018.
- Fowler, Martin. "Bliki: BoundedContext." *MartinFowler.com* (blog), January 15, 2014. <https://martinfowler.com/bliki/BoundedContext.html>.
- Fowler, Martin. "Bliki: MicroservicePrerequisites." *MartinFowler.com* (blog), August 28, 2014. <https://martinfowler.com/bliki/MicroservicePrerequisites.html>.
- Fried, Jason, and David Heinemeir Hansson. *Remote: Office Not Required*. NY: Crown Business, 2013.
- Gothelf, Jeff, and Josh Seiden. *Sense and Respond: How Successful Organizations Listen to Customers and Create New Products Continuously*. Boston, Massachusetts: Harvard Business Review Press, 2017.
- Greenleaf, Robert K. *The Servant as Leader*, Revised Edition. Atlanta, GA: The Greenleaf Center for Servant Leadership, 2015.
- "Guide: Understand Team Effectiveness." reWork website, <https://rework.withgoogle.com/guides/understanding-team-effectiveness/steps/define-team/>.

- Hall, Jon. "ITSM, DevOps, and Why Three-Tier Support Should Be Replaced with Swarming." *Medium*, December 17, 2016. https://medium.com/@JonHall_/itsm-devops-and-why-the-three-tier-structure-must-be-replaced-with-swarming-91e76ba22304.
- Hastie, Shane. "An Interview with Sam Guckenheimer on Microsoft's Journey to Cloud Cadence." *InfoQ*, October 17, 2014. <https://www.infoq.com/articles/agile2014-guckenheimer>.
- HBS Communications. "Collaborate on Complex Problems, but Only Intermittently." *Harvard Gazette* (blog), August 15, 2018. <https://news.harvard.edu/gazette/story/2018/08/collaborate-on-complex-problems-but-only-intermittently/>.
- Helfand, Heidi Shetzer. *Dynamic Reteaming: The Art and Wisdom of Changing Teams*. Heidi Helfand, 2018.
- Hoff, Todd. "Amazon Architecture." *High Scalability* (blog), September 18, 2007. <http://highscalability.com/blog/2007/9/18/amazon-architecture.html>.
- Holliday, Ben. "A 'Service-Oriented' Approach to Organisation Design." *FutureGov* (blog), September 25, 2018. <https://blog.wearefuturegov.com/a-service-oriented-approach-to-organisation-design-1e075be7f578>.
- Hoskins, Drew. "What Is It like to Be Part of the Infrastructure Team at Facebook?" *Quora*, last updated February 15, 2015. <https://www.quora.com/What-is-it-like-to-be-part-of-the-Infrastructure-team-at-Facebook>.
- Humble, Jez. "There's No Such Thing as a 'Devops Team'." *Continuous Delivery* (blog), October 19, 2012. <https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/>.
- Humble, Jez, and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Upper Saddle River, NJ: Addison Wesley, 2010.
- Humble, Jez, Joanne Molesky, and Barry O'Reilly. *Lean Enterprise: How High Performance Organizations Innovate at Scale*. Sebastopol, CA: O'Reilly Media, 2015.
- Ilgel, Daniel R., and John R. Hollenbeck. 'Effective Team Performance under Stress and Normal Conditions: An Experimental Paradigm, Theory and Data for Studying Team Decision Making in Hierarchical Teams with Distributed Expertise'. DTIC Document, 1993. <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA284683>.
- Ingles, Paul. "Convergence to Kubernetes." *Paul Ingles* (blog), June 18, 2018. <https://medium.com/@pingles/convergence-to-kubernetes-137ffa7ea2bc>.
- innolution. n.d. "Feature Team Definition | Innolution." Accessed October 14, 2018. <https://innolution.com/resources/glossary/feature-team>
- "DevOps Over Coffee—Adidas." YouTube video, 32:03, posted by IT Revolution, July 3, 2018. <https://www.youtube.com/watch?v=oOjdXeGp44E&feature=youtu.be&t=1071>.
- Jang, Sujin. "Cultural Brokerage and Creative Performance in Multicultural Teams." *Organization Science* 28 no. 6 (2017): 993–1009. <https://doi.org/10.1287/orsc.2017.1162>.
- Jay, Graylin, Joanne Hale, Randy Smith, David Hale, Nicholas Kraft, and Charles Ward. "Cyclo-matic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship." *Journal of Software Engineering & Applications* 2 (January): 137–143. <https://doi.org/10.4236/jsea.2009.23020>.
- John, Wolfgang. "DevOps for Service Providers—Next Generation Tools." *Ericsson Research Blog*. December 7, 2015. <https://www.ericsson.com/research-blog/cloud/devops-for-service-providers-next-generation-tools/>.

- Johnston, Joan H., Stephen M. Fiore, Carol Paris, and C. A. P. Smith. "Application of Cognitive Load Theory to Developing a Measure of Team Decision Efficiency." *Military Psychology* 3 (2003). <https://www.tandfonline.com/doi/abs/10.1037/h0094967>.
- Karlgård, Rich, and Michael S. Malone. *Team Genius: The New Science of High-Performing Organizations*. New York, NY: HarperBusiness, 2015.
- Kelly, Allan. *Business Patterns for Software Developers*. Chichester, UK: John Wiley & Sons, 2012.
- Kelly, Allan. "Conway's Law v. Software Architecture." Dzone.com (blog), March 14, 2013. <https://dzone.com/articles/conways-law-v-software>.
- Kelly, Allan. "Conway's Law & Continuous Delivery." SlideShare, posted by Allen Kelly, April 9, 2014, <https://www.slideshare.net/allankellynet/conways-law-continuous-delivery>.
- Kelly, Allan. "No Projects—Beyond Projects." *InfoQ*, December 5, 2014. <https://www.infoq.com/articles/kelly-beyond-projects>.
- Kelly, Allan. *Project Myopia: Why Projects Damage Software #NoProjects*. Allan Kelly: 2018.
- Kelly, Allan. "Return to Conway's Law." *Allan Kelly Associates* (blog), January 17, 2006. <https://www.allankellyassociates.co.uk/archives/1169/return-to-conways-law/>.
- Kersten, Mik. *Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework*. Portland, OR: IT Revolution Press, 2018.
- Kim, Gene, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR: IT Revolution Press, 2016.
- Kim, Dr. Kyung Hee, and Robert A. Pierce. "Convergent Versus Divergent Thinking." In *Encyclopedia of Creativity, Invention, Innovation and Entrepreneurship*, edited by Elias G. Carayannis, 245–250. New York: Springer, 2013.
- Kitagawa, Justin. "Platforms at Twilio: Unlocking Developer Effectiveness." *InfoQ*, October 18, 2018. <https://www.infoq.com/presentations/twilio-devops>
- Kitson, Jon. "Squad Health Checks." *Sky Betting & Gaming Technology* (blog), February 1, 2017. <https://technology.skybettingandgaming.com/2017/02/01/squad-health-checks/>.
- Kniberg, Henrik, and Anders Ivarsson. "Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds." *Crisp's Blog*. October 2012. <https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf>.
- Kniberg, Henrik. "Real-Life Agile Scaling." Presented at the Agile Tour Bangkok, Thailand, November 21, 2015. <http://blog.crisp.se/wp-content/uploads/2015/11/Real-life-agile-scaling.pdf>.
- Kniberg, Henrik. "Squad Health Check Model—Visualizing What to Improve." *Spotify Labs* (blog), September 16, 2014. <https://labs.spotify.com/2014/09/16/squad-health-check-model/>
- Knight, Pamela. "Acquisition Community Team Dynamics: The Tuckman Model vs. the DAU Model." *Proceedings from the 4th Annual Acquisition Research Symposium of the Naval Postgraduate School* (2007). <https://apps.dtic.mil/dtic/tr/fulltext/u2/a493549.pdf>.
- Kotter, John P. "Accelerate!" *Harvard Business Review*, November 1, 2012. <https://hbr.org/2012/11/accelerate>.
- Kramer, Staci D. "The Biggest Thing Amazon Got Right: The Platform." *Gigaom*, October 12, 2011. <https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/>.
- Laloux, Frédéric. *Reinventing Organizations: An Illustrated Invitation to Join the Conversation on Next-Stage Organizations*. Oxford, UK: Nelson Parker, 2016.

- Lane, Kim. "The Secret to Amazon's Success—Internal APIs." *API Evangelist* (blog), January 12, 2012. <http://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/>.
- Larman, Craig, and Bas Vodde. "Choose Feature Teams over Component Teams for Agility." *InfoQ*, July 15, 2008. <https://www.infoq.com/articles/scaling-lean-agile-feature-teams>.
- Larman, Craig, and Bas Vodde. *Large-Scale Scrum: More with LeSS*. Upper Saddle River, NJ: Addison-Wesley Professional, 2016.
- Leffingwell, Dean. "Feature Teams vs. Component Teams (Continued)." *Scaling Software Agility* (blog), May 2, 2011. <https://scalingsoftwareagility.wordpress.com/2011/05/02/feature-teams-vs-component-teams-continued/>.
- Leffingwell, Dean. "Organizing at Scale: Feature Teams vs. Component Teams—Part 3." *Scaling Software Agility* (blog), July 22, 2009. <https://scalingsoftwareagility.wordpress.com/2009/07/22/organizing-agile-at-scale-feature-teams-versus-component-teams-part-3/>.
- Leffingwell, Dean. *Scaling Software Agility: Best Practices for Large Enterprises*. Upper Saddle River, NJ: Addison-Wesley Professional, 2007.
- Lencioni, Patrick M. *The Five Dysfunctions of a Team: A Leadership Fable*. San Francisco, CA: John Wiley & Sons, 2002.
- Leveson, Nancy G. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press, 2017.
- Levina, Natalia, and Emmanuelle Vaast. "The Emergence of Boundary Spanning Competence in Practice: Implications for Information Systems' Implementation and Use." *MIS Quarterly* 29 no. 2 (June 2005): 335–363. <https://papers.ssrn.com/abstract=1276022>.
- Lewis, James. "Microservices and the Inverse Conway Manoeuvre—James Lewis." YouTube video, 57:57, posted by NDC Conferences, February 16, 2017. <https://www.youtube.com/watch?v=uamh7xppO3E>.
- Lim, Beng-Chong, and Katherine J. Klein. "Team Mental Models and Team Performance: A Field Study of the Effects of Team Mental Model Similarity and Accuracy." *Journal of Organizational Behavior* 27, no. 4 (June 1, 2006): 403–418. <https://doi.org/10.1002/job.387>.
- Linders, Ben. "Scaling Teams to Grow Effective Organizations." *InfoQ*, August 11, 2016. <https://www.infoq.com/news/2016/08/scaling-teams>.
- Long, Josh. "GARY (Go Ahead, Repeat Yourself)." Tweet @starbuxman, May 25, 2016. <https://twitter.com/starbuxman/status/735550836147814400>.
- Lowe, Steven A. "How to Use Event Storming to Achieve Domain-Driven Design." *TechBeacon*, October 15, 2015. <https://techbeacon.com/introduction-event-storming-easy-way-achieve-domain-driven-design>.
- Luo, Jiao, Andrew H. Van de Ven, Runtian Jing, and Yuan Jiang. "Transitioning from a Hierarchical Product Organization to an Open Platform Organization: A Chinese Case Study." *Journal of Organization Design* 7 (January): 1. <https://doi.org/10.1186/s41469-017-0026-x>.
- MacCormack, Alan, John Rusnak, and Carliss Y. Baldwin. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." *Management Science* 52, no. 7 (2006): 1015–1030. <https://doi.org/10.1287/mnsc.1060.0552>.
- MacCormack, Alan, Carliss Y. Baldwin, and John Rusnak. "Exploring the Duality Between Product and Organizational Architectures: A Test of the 'Mirroring' Hypothesis." *Research Policy* 41, no. 8 (October 2012): 1309–1024. <http://www.hbs.edu/faculty/Pages/item.aspx?num=43260>.

- Malan, Ruth. "Conway's Law." *TraceintheSand.com* (blog), February 13, 2008. <http://traceinthesand.com/blog/2008/02/13/conways-law/>.
- Manns, Mary Lynn, and Linda Rising. *Fearless Change: Patterns for Introducing New Ideas*. Boston, MA: Addison Wesley, 2004.
- Marshall, Bob. "A Team Is Not a Group of People Who Work Together. A Team Is a Group of People Who Each Put the Team before Themselves." Tweet, @flowchainsensei, October 29, 2018. <https://twitter.com/flowchainsensei/status/1056838136574152704>.
- McChrystal, General Stanley, David Silverman, Tatum Collins, and Chris Fussell. *Team of Teams: New Rules of Engagement for a Complex World*. New York, NY: Portfolio Penguin, 2015.
- Meadows, Donella. *Leverage Points: Places to Intervene in a System*. Hartland, VT: Sustainability Institute, 1999. http://donellameadows.org/wp-content/userfiles/Leverage_Points.pdf.
- "Microservices: Organizing Large Teams for Rapid Delivery." SlideShare, posted by Pivotal, August 10, 2016. <https://www.slideshare.net/Pivotal/microservices-organizing-large-teams-for-rapid-delivery>.
- Mihaljov, Timo. "Having a Dedicated DevOps Person Who Does All the DevOpsing Is like Having a Dedicated Collaboration Person Who Does All the Collaborating." Tweet. @noidi. April 14, 2017. <https://twitter.com/noidi/status/852879869998501889>.
- Miller, G. A. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." *Psychological Review* 63 no. 2 (1956): 81–97.
- Minick, Eric. "The Goal for a 'DevOps Team' Should Be to Put Itself out of Business by Enabling the Rest of the Org." Tweet, @ericminick, October 8, 2014. <https://twitter.com/ericminick/status/517335119330172930>.
- Minick, Eric, and Curtis Yanko. "Creating a DevOps Team That Isn't Evil." SlideShare, posted by IBM Urban Code Products, March 5, 2015. <http://www.slideshare.net/Urbancode/creating-a-devops-team-that-isnt-evil>.
- Mole, David. "Drive: How We Used Daniel Pink's Work to Create a Happier, More Productive Work Place." *InfoQ*, September 10, 2015. <https://www.infoq.com/articles/drive-productive-workplace>.
- Morgan-Smith, Victoria, and Matthew Skelton. *Internal Tech Conferences*. Leeds, UK: Conflux Digital, 2019.
- Morris, Kief. *Infrastructure as Code: Managing Servers in the Cloud*. Sebastopol, CA: O'Reilly Media, 2016.
- Munns, Chris. "Chris Munns, DevOps @ Amazon: Microservices, 2 Pizza Teams, & 50 Million Deploys per Year." SlideShare.net, posted by TriNimbus, May 6, 2016. <http://www.slideshare.net/TriNimbus/chris-munns-devops-amazon-microservices-2-pizza-teams-50-million-deploys-a-year>.
- Murphy, Niall. "What is 'Site Reliability Engineering'?" Landing.Google.com, <https://landing.google.com/sre/interview/ben-treynor.html>.
- Murphy, Niall and Ben Treynor. "What is 'Site Reliability Engineering'?" Landing.Google.com (blog), accessed March 21, 2019. <https://landing.google.com/sre/interview/ben-treynor.html>.
- Narayan, Sriram. *Agile IT Organization Design: For Digital Transformation and Continuous Delivery*. New York: Addison-Wesley Professional, 2015.

- Neumark, Peter. “DevOps & Product Teams—Win or Fail?” *InfoQ*, June 29, 2015. <https://www.infoq.com/articles/devops-product-teams>.
- Netflix Technology Blog. “Full Cycle Developers at Netflix—Operate What You Build.” *Medium.com*, May 17, 2018, <https://medium.com/netflix-techblog/full-cycle-developers-at-netflix-a08c31f83249>.
- Netflix Technology Blog. “The Netflix Simian Army.” Netflix TechBlog, July 19, 2011. <https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116>.
- Newman, Sam. *Building Microservices: Design Fine-Grained Systems*. Sebastopol, CA: O’Reilly Media, 2015.
- Newman, Sam. “Demystifying Conway’s Law.” ThoughtWorks (blog) June 30, 2014. <https://www.thoughtworks.com/insights/blog/demystifying-conways-law>.
- Nygard, Michael. “The Perils of Semantic Coupling—Wide Awake Developers.” MichaelNygard.com (blog), April 29, 2015. <http://michaelnygard.com/blog/2015/04/the-perils-of-semantic-coupling/>.
- Nygard, Michael T. *Release It! Design and Deploy Production-Ready Software*, 2nd edition. Raleigh, North Carolina: O’Reilly, 2018.
- O’Connor, Debra L., and Tristan E. Johnson. “Understanding Team Cognition in Performance Improvement Teams: A Meta-Analysis of Change in Shared Mental Models.” *Proceedings of the Second International Conference on Concept Mapping* (2006). <https://pdfs.semanticscholar.org/4106/3eb1567e630a35b4f33f281a6bb9d193ddf5.pdf>.
- O’Dell, Chris. “You Build It, You Run It (Why Developers Should Also Be on Call).” Skelton Thatcher.com (blog), October 18, 2017. <https://skeltonthatcher.com/blog/build-run-developers-also-call/>.
- Overeem, Barry. “How I Used the Spotify Squad Health Check Model—Barry Overeem—The Liberators.” BarryOvereem.com (blog), August 7, 2015. <http://www.barryovereem.com/how-i-used-the-spotify-squad-health-check-model/>.
- Pais, Manuel. “Damon Edwards: DevOps is an Enterprise Concern” *InfoQ*, May 31, 2014. <https://www.infoq.com/interviews/interview-damon-edwards-qcon-2014>.
- Pais, Manuel. “Prezi’s CTO on How to Remain a Lean Startup after 4 Years.” *InfoQ*, October 5, 2012. <https://www.infoq.com/news/2012/10/Prezi-lean-startup>.
- Pais, Manuel, and Matthew Skelton. “The Divisive Effect of Separate Issue Tracking Tools.” *InfoQ*, March 22, 2017. <https://www.infoq.com/articles/issue-tracking-tools>.
- Pais, Manuel, and Matthew Skelton. “Why and How to Test Logging.” *InfoQ*, October 29, 2016. <https://www.infoq.com/articles/why-test-logging>.
- Pearce, Jo. “Day 3: Managing Cognitive Load for Team Learning.” *12 Devs of Xmas* (blog), December 28, 2015. <http://12devsofxmas.co.uk/2015/12/day-3-managing-cognitive-load-for-team-learning/>.
- Pearce, Jo. “Hacking Your Head: Managing Information Overload (Extended).” SlideShare, posted by Jo Pearce, April 29, 2016. <https://www.slideshare.net/JoPearce5/hacking-your-head-managing-information-overload-extended>.
- Perri, Melissa. *Escaping the Build Trap: How Effective Product Management Creates Real Value*. Sebastopol, CA: O’Reilly, 2018.
- Pflaeging, Niels. *Organize for Complexity: How to Get Life Back Into Work to Build the High-Performance Organization*, 1st edition. Germany: BetaCodex Publishing, 2014.

- Pflaeging, Niels. "Org Physics: The 3 Faces of Every Company." *Niels Pflaeging* (blog), March 6, 2017. <https://medium.com/@NielsPflaeging/org-physics-the-3-faces-of-every-company-df16025f65f8>.
- Phillips, Amy. "Testing Observability." *InfoQ*, April 5, 2018. <https://www.infoq.com/presentations/observability-testing>.
- Pink, Daniel. *Drive: The Surprising Truth About What Motivates Us*. New York: Riverhead Books, 2009.
- Raymond, Eric. *The New Hacker's Dictionary*, 3rd Edition. Boston, MA: MIT Press, 1996.
- Reed, J. Paul. "Blameless Postmortems Don't Work. Be Blame-Aware but Don't Go Negative." *TechBeacon*, March 22, 2016. <https://techbeacon.com/blameless-postmortems-dont-work-heres-what-does>.
- Reinertsen, Donald. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Redondo Beach, CA: Celeritas Publishing, 2009.
- Rensin, Dave. "Introducing Google Customer Reliability Engineering." Google Cloud Blog, October 10, 2016. <https://cloud.google.com/blog/products/gcp/introducing-a-new-era-of-customer-support-google-customer-reliability-engineering/>.
- Roberts, John. *The Modern Firm: Organizational Design for Performance and Growth*. Oxford: Oxford University Press, 2007.
- Robertson, Brian J. *Holocracy: The New Management System for a Rapidly Changing World*. NY: Henry Holt, 2015.
- Rock, David, and Heidi Grant. *Why Diverse Teams Are Smarter*. Cambridge, MA: Harvard Business Review, 2016.
- Rother, Mike. *Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results*. New York: McGraw-Hill Education, 2009.
- Rozovsky, Julia. "Re:Work—The Five Keys to a Successful Google Team." *re:Work* (blog), November 17, 2015. <https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/>.
- Rubin, Kenneth S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Upper Saddle River, NJ: Addison Wesley, 2012.
- Rummler, Geary, and Alan Brache. *Improving Performance: How to Manage the White Space on the Organization Chart*, 3rd edition. San Francisco, CA: Jossey-Bass, 2013.
- Salas, Eduardo, and Stephen M. Fiore, eds. *Team Cognition: Understanding the Factors That Drive Process and Performance*. Washington, DC: American Psychological Association, 2004.
- Scholtes, Ingo, Pavlin Mavrodiev, and Frank Schweitzer. "From Aristotle to Ringelmann: A Large-Scale Analysis of Team Productivity and Coordination in Open Source Software Projects." *Empirical Software Engineering* 21 no. 2 (2016): 642–683. <https://doi.org/10.1007/s10664-015-9406-4>.
- Schotkamp, Tom, and Martin Danoesastro. "HR's Pioneering Role in Agile at ING." *BCG* (blog), June 1, 2018. <https://www.bcg.com/en-gb/publications/2018/human-resources-pioneering-role-agile-ing.aspx>.
- Schwartz, Mark, Jason Cox, Jonathan Snyder, Mark Rendell, Chivas Nambiar, and Mustafa Kapadia. *Thinking Environments: Evaluating Organization Models for DevOps to Accelerate*. Portland, OR: IT Revolution Press, 2016.
- Seiter, Courtney. "We've Changed Our Product Team Structure 4 Times: Here's Where We Are Today." *Buffer* (blog), October 20, 2015. <https://open.buffer.com/product-team-evolution/>.

- Shibata, Kenichi. “How to Build a Platform Team Now! The Secrets to Successful Engineering.” *Hacker Noon* (blog), September 29, 2018. <https://hackernoon.com/how-to-build-a-platform-team-now-the-secrets-to-successful-engineering-8a9b6a4d2c8>.
- Simenon, Stefan, and Wiebe de Roos. “Transforming CI/CD at ABN AMRO to Accelerate Software Delivery and Improve Security.” SlideShare, posted by DevOps.com, March 27, 2018. <https://www.slideshare.net/DevOpsWebinars/transforming-cicd-at-abn-amro-to-accelerate-software-delivery-and-improve-security>.
- Sinha, Harsh. “Harsh Sinha on Building Culture at TransferWise.” *InfoQ*, February 19, 2018. <https://www.infoq.com/podcasts/Harsh-Sinha-transferwise-building-culture>.
- Skelton, Matthew. “How Different Team Topologies Influence DevOps Culture.” *InfoQ*, September 2, 2015. <https://www.infoq.com/articles/devops-team-topologies>.
- Skelton, Matthew. “How to Find the Right DevOps Tools for Your Team.” *TechBeacon*, 2018. <https://techbeacon.com/how-find-right-devops-tools-your-team>.
- Skelton, Matthew. “Icebreaker for Agile Retrospectives—Empathy Snap.” *MatthewSkelton.net* (blog), November 15, 2012. <http://empathysnap.com/>.
- Skelton, Matthew. *Tech Talks for Beginners*. Leeds, UK: Conflux Digital, 2018.
- Skelton, Matthew. “What Team Structure Is Right for DevOps to Flourish?” *Matthew Skelton.net* (blog), October 22, 2013. <https://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/>.
- Skelton, Matthew. “Your Team’s API Includes: - Code: REST Endpoints, Libraries, Clients, UI, Etc.—Wiki / Docs—Especially ‘How To’ Guides—Your Approach to Team Chat Tools (Slack / Hipchat)—Anything Else Which Other Teams Need to Use to Interact with Your Team It’s Not Just about Code. #DevEx.” Tweet, @matthewpskelton, July 25, 2018. <https://twitter.com/matthewpskelton/status/1022111880423395329>.
- Skelton, Matthew, and Rob Thatcher. *Team Guide to Software Operability*. Leeds, UK: Conflux Books, 2016.
- Skulmowski, Alexander, and Rey, Günter Daniel. “Measuring Cognitive Load in Embodied Learning Settings.” *Frontiers in Psychology* 8 (August 2, 2017). <https://doi.org/10.3389/fpsyg.2017.01191>.
- Smith, Steve, and Matthew Skelton, eds. *Build Quality In*. Leeds, UK: Conflux Digital, 2015.
- Snowden, Dave. “The Rule of 5, 15 & 150.” *Cognitive Edge* (blog), December 10, 2006. <http://cognitive-edge.com/blog/logn-0-093-3-389-logcr-1-r20-764-t3410-35-p0-001/>.
- Sosa, Manuel E., Steven D. Eppinger, and Craig M. Rowles. “The Misalignment of Product Architecture and Organizational Structure in Complex Product Development.” *Management Science* 50 no. 12 (December 2004): 1674–1689.
- Stanford, Naomi. *Guide to Organisation Design: Creating High-Performing and Adaptable Enterprises* (Economist Books), 2nd Edition. London: Economist Books, 2015.
- Stomppff, Guido. “Facilitating Team Cognition: How Designers Mirror What NPD Teams Do.” *ResearchGate*, September 2012. https://www.researchgate.net/publication/254831689_Facilitating_Team_Cognition_How_designers_mirror_what_NPD_teams_do.
- Strode, Diane E., Sid L. Huff, Beverley Hope, and Sebastian Link. “Coordination in Co-Located Agile Software Development Projects.” *Journal of Systems and Software, Special Issue: Agile Development* 85, no. 6 (June 1, 2012): 1222–38. <https://doi.org/10.1016/j.jss.2012.02.017>.

- Sussna, Jeff. *Designing Delivery: Rethinking IT in the Digital Service Economy*. Sebastopol, CA: O'Reilly Media, 2015.
- Sweller, John. "Cognitive Load During Problem Solving: Effects on Learning." *Cognitive Science* 12 no. 2 (1988): 257–285.
- Sweller, John. "Cognitive Load Theory, Learning Difficulty, and Instructional Design." *Learning and Instruction* 4 (1994): 295–312.
- "System Team." Scaled Agile Framework website, last updated October 5, 2018. <https://www.scaledagileframework.com/system-team/>.
- Tuckman, Bruce W. "Developmental Sequence in Small Groups." *Psychological Bulletin* 63 no. 6 (1965): 384–399. <https://doi.org/10.1037/h0022100>.
- Tune, Nick. "Domain-Driven Architecture Diagrams." *Nick Tune's Tech Strategy Blog*, August 15, 2015. <https://medium.com/nick-tune-tech-strategy-blog/domain-driven-architecture-diagrams-139a75acb578>.
- Tune, Nick, and Scott Millett. *Designing Autonomous Teams and Services*. Sebastopol, CA: O'Reilly Media, 2017.
- Urquhart, James. "Communications and Conway's Law." *Digital Anatomy* (blog), September 28, 2016. <https://medium.com/digital-anatomy/communications-and-conways-law-6a1a9deae32>.
- Urquhart, James. "IT Operations in a Cloudy World." *CNET*, September 15, 2010. <https://www.cnet.com/news/it-operations-in-a-cloudy-world/>.
- Wardley, Simon. "An Introduction to Wardley 'Value Chain' Mapping." *CIO UK*, March 19, 2015. <https://www.cio.co.uk/it-strategy/introduction-wardley-value-chain-mapping-3604565/>.
- Wastell, Katherine. "What We Mean When We Talk about Service Design at the Co-Op." Co-Op Digital Blog, October 25, 2018. <https://digitalblog.coop.co.uk/2018/10/25/what-we-mean-when-we-talk-about-service-design-at-the-co-op/>.
- Webber, Emily. *Building Successful Communities of Practice*. San Francisco, CA: Blurb, 2018.
- Weinberg, Gerald M. *An Introduction to General Systems Thinking, 25th Silver Anniversary Edition*. New York: Dorset, 2001.
- Wiener, Norbert. *Cybernetics: Or Control and Communication in the Animal and the Machine*, 2nd edition. Cambridge, Mass: MIT Press, 1961.
- Westrum, R. 2004. "A Typology of Organisational Cultures." *Quality & Safety in Health Care* 13 Suppl. 2 (1961): ii22–27. <https://doi.org/10.1136/qshc.2003.009522>.
- "What Team Structure is Right for DevOps to Flourish?" DevOpsTopologies.com, accessed March 21, 2019. <http://web.devopstopologies.com>.
- Wiley, Evan. "Scaling XP Through Self-Similarity at Pivotal Cloud Foundry." *Agile Alliance* (blog), July 28, 2018. <https://www.agilealliance.org/resources/experience-reports/scaling-xp-through-self-similarity-at-pivotal-cloud-foundry/>.
- Womack, James P., and Daniel T. Jones. *Lean Thinking: Banish Waste and Create Wealth In Your Corporation*. NY: Simon & Schuster/Free Press, 2003.
- Zambonelli, Franco. "Toward Sociotechnical Urban Superorganisms." *Computer*, 2012. <http://spartan.ac.brocku.ca/~tkennedy/COMM/Zambonelli2012.pdf>.

NOTES

Foreword

1. Conway, “How Do Committees Invent?”

Preface

1. Skelton, “What Team Structure Is Right for DevOps to Flourish?”
2. Skelton, “How Different Team Topologies Influence DevOps Culture.”

Chapter 1

1. Schwartz et al., *Thinking Environments*, 21.
2. Pflaeging, *Organize for Complexity*, 34–41.
3. Pflaeging, *Organize for Complexity*.
4. Laloux, *Reinventing Organizations*; Robertson, *Holocracy*.
5. Stanford, *Guide to Organisation Design*, 14–16.
6. Conway, “How do Committees Invent?”, 31.
7. Conway, “How do Committees Invent?”; Kelly, “Conway’s Law & Continuous Delivery.”
8. Kelly, “Conway’s Law v. Software Architecture.”
9. Raymond, *The New Hacker’s Dictionary*, 124.
10. Lewis, “Microservices and the Inverse Conway.”
11. Pink, *Drive*, 49.

Chapter 2

1. “DevOps Over Coffee – Adidas,” Fernando Cornago, person email communication with the authors, March 2019.
2. MacCormack et al., “Exploring the Structure of Complex Software Designs,” 1015–1030; MacCormack et al., “Exploring the Duality Between Product and Organizational Architectures,” 1309–1024.
3. Sosa et al., “The Misalignment of Product Architecture and Organizational Structure in Complex Product Development,” 1674–1689.
4. Malan, “Conway’s Law.”
5. Conway, “How do Committees Invent?” 28.
6. Forsgren et al., *Accelerate*, 63.
7. Nygard, *Release It!*, 4.
8. MacCormack et al., “Exploring the Structure of Complex Software Designs.”
9. Roberts, *The Modern Firm*, 190.
10. Reinertsen, *The Principles of Product Development Flow*, 257.
11. Malan, “Conway’s Law.”
12. Kelly, “Return to Conway’s Law.”

3. Kniberg and Ivarsson, “Scaling Agile @ Spotify.”
4. Kniberg and Ivarsson, “Scaling Agile @ Spotify.”
5. Forsgren et al., *Accelerate*, 63.
6. Skelton, “What Team Structure Is Right for DevOps to Flourish?”
7. John, “DevOps for Service Providers—Next Generation Tools.”
8. Hastie, “An Interview with Sam Guckenheimer on Microsoft’s Journey to Cloud Cadence.”
9. Ben Treynor, as quoted in Niall Murphy, “What is ‘Site Reliability Engineering’?”
10. Dogan, “The SRE Model.”
11. Rensin, “Introducing Google Customer Reliability Engineering.”
12. Netflix Technology Blog, “Full Cycle Developers at Netflix—Operate What You Build.”
13. DeGrandis, *Making Work Visible*, 82.
14. Strode and Huff, “A Taxonomy of Dependencies in Agile Software Development.”
15. Pulak Agrawal, personal communication with the authors, March 2019.
16. Pulak Agrawal, personal communication with the authors, March 2019.

Chapter 5

1. Luo et al., “Transitioning from a Hierarchical Product Organization to an Open Platform Organization.”
2. Reinertsen, *The Principles of Product Development Flow*, 265.
3. Lane, “The Secret to Amazon’s Success—Internal APIs;” Hoff, “Amazon Architecture.”
4. Crawford, “Amazon’s ‘Two-Pizza Teams;” Munns, “Chris Munns, DevOps @ Amazon.”
5. Kramer, “The Biggest Thing Amazon Got Right.”
6. Sussna, *Designing Delivery*, 148.
7. Pink, *Drive*, 49.
8. Eckstein, “Architecture in Large Scale Agile Development,” 21–29.
9. Robert Greenleaf, *The Servant as Leader*.
10. DeMarco and Lister, *Peopleware*, 212.
11. Webber, *Building Successful Communities of Practice*, 11.
12. Bottcher, “What I Talk About When I Talk About Platforms.”
13. Eckstein, *Agile Development in the Large*, 53.
14. Neumark, “DevOps & Product Teams—Win or Fail?”
15. Reinertsen, *The Principles of Product Development Flow*, 292.
16. Womack and Jones, *Lean Thinking*.
17. Urquhart, “IT Operations in a Cloudy World.”
18. Kniberg, “Real-Life Agile Scaling.”
19. Kelly, *Business Patterns for Software Developers*, 88–89.
20. Conway, “Toward Simplifying Application Development, in a Dozen Lessons.”
21. Shibata, “How to Build a Platform Team Now!”
22. Shibata, “How to Build a Platform Team Now!”
23. Beer, *Brain of the Firm*, 238.
24. Shibata, “How to Build a Platform Team Now!”
25. Hall, “ITSM, DevOps, and Why Three-Tier Support Should Be Replaced with Swarming.”
26. Forsgren et al., *Accelerate*, 68.

Chapter 6

1. Forsgren et al., *Accelerate*, 63.
2. Forsgren et al., *Accelerate*, 66
3. Bernstein and Turban, “The Impact of the ‘Open’ Workspace on Human Collaboration.”
4. Evans, *Domain-Driven Design*.
5. Fowler, “Bliki: BoundedContext.”

6. Tune and Millett, *Designing Autonomous Teams and Services*, 38.
7. Nygard, "The Perils of Semantic Coupling."
8. Helfand, *Dynamic Reteaming*, 203.
9. Hering, *DevOps for the Modern Enterprise*, 45.
10. Phillips, "Testing Observability."

Chapter 7

1. Bernstein et al., "How Intermittent Breaks in Interaction Improve Collective Intelligence," 8734–8739.
2. Rother, *Toyota Kata*, 236.
3. Kim and Pierce, "Convergent Versus Divergent Thinking," 245–250.
4. Urquhart, "Communications and Conway's Law."
5. Betz, *Managing Digital*, 253.
6. Burgess, *Thinking in Promises*, 105.
7. Reinertsen, *The Principles of Product Development Flow*, 233.
8. Malan, "Conway's Law."
9. Kelly, "Return to Conway's Law."
10. Helfand, *Dynamic Reteaming*, 121; Wiley, as quoted in Helfand, *Dynamic Reteaming*, 121.
11. Helfand, *Dynamic Reteaming*, 13.
12. Reinertsen, *The Principles of Product Development Flow*, 254.

Chapter 8

1. Forsgren et al., *Accelerate*, 63.
2. Ingles, "Convergence to Kubernetes."
3. Ingles, "Convergence to Kubernetes,"
4. Sussna, *Designing Delivery*, 61.
5. Kotter, "Accelerate!"
6. Drucker, *The Daily Drucker*, 291.
7. Stanford, *Guide to Organisation Design*, 17.
8. Narayan, *Agile IT Organization Design*, 65.
9. Kim et al., *The DevOps Handbook*, 11.
10. Sussna, *Designing Delivery*, 58.
11. Narayan, *Agile IT Organization Design*, 31.

Conclusion

1. Conway, "How do Committees Invent?" 31.
2. Manns and Rising, *Fearless Change*.

13. Stanford, *Guide to Organisation Design*, 4.
14. Sosa et al., “The Misalignment of Product Architecture.”
15. Cohn, “Nine Questions to Assess Scrum Team Structure.”
16. Kniberg, “Real-Life Agile Scaling.”

Chapter 3

1. Driskell and Salas, “Collective Behavior and Team Performance,” 277–288.
2. McChrystal et al., *Team of Teams*, 94.
3. Rozovsky, “Re:Work—The Five Keys to a Successful Google Team.”
4. Crawford, At opening quotes. “Amazon’s ‘Two-Pizza Teams.’”
5. Dunbar, “Neocortex Size as a Constraint on Group Size in Primates,” 469–493.
6. Snowden, “The Rule of 5, 15 & 150;” Dunbar, *How Many Friends Does One Person Need?*; Bennett, “The Dunbar Number, From the Guru of Social Networks;” Burgess, *Thinking in Promises*, 87.
7. Snowden, “The Rule of 5, 15 & 150;” Karlgaard and Malone, *Team Genius*, 201–205.
8. Lewis, “Microservices and the Inverse Conway Manoeuvre.”
9. Munns, “Chris Munns, DevOps @ Amazon.”
10. Brooks, *The Mythical Man-Month*.
11. Tuckman, “Developmental Sequence in Small Groups,” 384–399.
12. Kelly, *Project Myopia*, 72.
13. Helfand, *Dynamic Reteaming*, 123.
14. Knight, “Acquisition Community Team Dynamics.”
15. Humble et al., *Lean Enterprise*, 37.
16. Driskell and Salas, “Collective Behavior and Team Performance;” Rock and Grant, *Why Diverse Teams Are Smarter*.
17. Jang, “Cultural Brokerage and Creative Performance in Multicultural Teams,” 993–1009; Carayon, “Human Factors of Complex Sociotechnical Systems,” 525–535.
18. DeMarco and Lister, *Peopleware*, 156.
19. Stanford, *Guide to Organisation Design*, 287.
20. Deming, *Out of the Crisis*, 22.
21. Roberts, *The Modern Firm*, 277.
22. Sweller, “Cognitive Load During Problem Solving: Effects on Learning,” 257–285.
23. Pearce, “Day 3: Managing Cognitive Load for Team Learning;” Pearce, “Hacking Your Head.”
24. Driskell et al., “Does Stress Lead to a Loss of Team Perspective,” 300.
25. Jay et al., “Cyclomatic Complexity and Lines of Code,” 137–143.
26. MacChrystal et al., *Team of Teams*, 94.
27. Lim and Klein, “Team Mental Models and Team Performance,” 403–418.
28. Evan Wiley, as quoted in Helfand, *Dynamic Reteaming*, 121.
29. Jeff Bezos, as quoted in Lane, “The Secret to Amazon’s Success.”
30. Axelrod, *Complexity of Cooperation*; Burgess, *Thinking in Promises*, 73.
31. Kniberg and Ivarsson, “Scaling Agile @ Spotify.”
32. Kniberg and Ivarsson, “Scaling Agile @ Spotify.”
33. Forsgren et al., *Accelerate*, 181.
34. Jeremy Brown, personal communication with the authors, March 2019.
35. Doorley and Witthoft, *Make Space*, 16.
36. Fried and Hansson, *Remote*, 91.

Chapter 4

1. Stanford, *Guide to Organisation Design*, 3.
2. Kniberg and Ivarsson, “Scaling Agile @ Spotify.”

INDEX

A

Accelerate: The Science of Dev Ops (Forsgren, Humble, & Kim), 18, 64, 86, 112, 114, 154
Accenture, 75–76
“Acquisition Community Team Dynamics: The Tuckman Model vs. the DAU Model” (Knight), xx
ad hoc team design, 62
adaptive structuration theory, xix
Adidas, 16
adoption of new practices, 155–159
Agile IT Organization Design (Narayan), 173
Agrawal, Pulak, 75–76
Amazon, 32, 35, 82–83, 112
anti-patterns, 62
Antunes, Miguel, 11
APIs
 defined, 187
 effective, 148
 team, 47–56
application monolith, 113, 187
Auto Trader, 53–55, 97–99
awkward team interactions, 150–151
AWS, 49
Axelrod, Robert, 49
Azure, 69, 101

B

Balena.io, 101
basic team organization, 146–148
BCG Digital Ventures, 88–90
Beer, Stafford, 103
benched bay approach, 51
Bertilsson, Albert, 47
Bezos, Jeff, 49, 82–83
Boone, Mary, xxi
Borland Delphi, 101
Bottcher, Evan, 92
bottlenecks, 11–12
boundaries, 39–47
 domain limitations, 42–45
 misplaced, 150–151
 relative domain complexity, 41–42
 responsibility restriction, 39–41
 software boundary size, 45–47
 team-first, 111–126

bounded context, 115–116, 187
Brain of the Firm (Beer), 103
Brooks, Fred, 35
Brooks’s law, 40, 187
Brown, Jeremy, 53
Burgess, Mark, 49, 142
business as usual teams, 173–174
business domain bounded context, 115–116
business process management, 169

C

capabilities
 missing, 150–151
 self-service, 69
capability gaps, 184–185
“Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory” (DeSanctis and Poole), xxi
case studies
 complicated-subsystems teams, 94–95, 97–99
 DevOps Topologies, 75–77
 enabling teams, 88–90
 fracture planes, 121–125
 organizational sensing, 154–155, 157–159, 162–164
 software boundaries, 121–125
 static topologies, 75–77
 stream-aligned teams, 82–83
 team APIs, 50–52, 53–55
 team interaction modes, 146–147
 team types, 82–83, 88–90
 team-first boundaries, 121–125
 team-first thinking, 50–52, 53–55
CDL, 50–52
change cadence, 117
cloud teams, 69–70
cognitive load, 11–12, 39–47
 defined, 187
 domain identification, 43
 domain limitations, 42–45
 ecosystem tuning, 46
 extraneous, 40, 187
 “Eyes On, Hands Off,” 46
 germane, 40, 188
 heuristics for domain assignment, 43
 intrinsic, 40, 188

- cognitive load (*continued*)
 - relative domain complexity, 41–42
 - responsibility restriction, 39–41
 - software boundary size, 45–47
 - types of, 39–40
 - Cohn, Mike, 24
 - collaboration mode, 9, 133, 135–137, 153–155
 - defined, 187
 - and evolution of team topologies, 159–161
 - and reverse Conway maneuver, 147–148
 - right amount of collaboration, 153–154
 - team behaviors for, 142–143
 - and viable X-as-a-service interactions, 149
 - commodity systems, 18
 - communication
 - focused, 24–26
 - inter-team, 25–26, 27
 - paths, 17
 - structures, 4–8
 - tool choice and, 27
 - unnecessary, 24–26
 - communities of practice vs. enabling teams, 90
 - compatible support systems, 69
 - complicated-subsystems teams, 9, 28, 91–92
 - case study, 94–95, 97–99
 - defined, 187
 - expected behaviors, 94
 - platform composition, 96–97
 - component teams, 28, 105–106
 - continuous delivery (CD), 11
 - continuous integration (CI), 11
 - converting teams
 - component teams to platform teams, 105–106
 - converting architecture and architects, 109
 - converting common to fundamental team topologies, 104–109
 - infrastructure teams to platform teams, 105
 - stream-aligned teams, 104
 - support teams, 107–109
 - tooling teams to enabling teams, 106–107
 - Conway, Mel, 9–10
 - Conway maneuver, inverse, 10, 18
 - Conway maneuver, reverse, 10, 18–21, 147–148, 188
 - Conway's law, xxii, 9–11, 15–29, 180–181
 - commodity systems, 18
 - communication paths, 17
 - complicated-subsystems teams, 28
 - component teams, 28
 - cross-team testing, 23
 - database administrator team, 18–19
 - defined, 187
 - focused communication, 24–26
 - high cohesion, 22
 - homomorphic force, 19–20
 - inter-team communication, 25–26, 27
 - inverse Conway maneuver, 10, 18
 - log-aggregation tools, 27–28
 - loose coupling, 22
 - modern version of, 17
 - monolithic shared databases, 16
 - naive uses of, 26–28
 - organization design, 16–17, 23–24
 - reorganizations, 28
 - reverse Conway maneuver, 10, 18–21, 147–148, 188
 - software architecture as flows of change, 23
 - team assignments, 22
 - team-scoped flow, 21–23
 - tool choice and communication, 27
 - understanding and using, 15–17
 - unnecessary communication, 24–26
 - version compatibility, 22
 - Cornago, Fernando, 16
 - coupled releases, 113
 - credit reference agencies, 76–77
 - cross-team testing, 23
 - Cybernetics: Or Control and Communication in the Animal and the Machine* (Wiener), xxi
- D**
- Daly, Damien, 121–123
 - database administrator team, 18–19
 - DeMarco, Tom, 38
 - Deming, W. Edwards, 38
 - dependencies, 74–75
 - DeSanctis, Gerardine, xxi
 - Designing Autonomous Teams and Services* (Tune & Millett), 116
 - Designing Delivery* (Sussna), 85, 172
 - “Developmental Sequence in Small Groups” (Tuckman), xxii
 - DevOps for the Modern Enterprise* (Hering), 120
 - The DevOps Handbook* (Kim), 166, 172
 - DevOps Topologies, 75–77
 - case study, 75–77
 - catalog, 66
 - credit reference agencies, 76–77
 - healthcare organizations, 75–76
 - DevOps Topologies catalog, 66
 - divergent thinking, 136, 138
 - diversity, 38
 - domain assignment, heuristics for, 43
 - domain complexity, xxi, 187
 - domain identification, 43
 - domain limitations, 42–45
 - Domain-Driven Design* (Evans), 115
 - Doorley, Scott, 53
 - Drexler, Allan, xxii
 - Drucker, Peter, 170
 - Dunbar, Robin, 32
 - Dunbar's number, 32, 187
 - Dynamic Reteaming* (Helfand), 48, 118, 150
- E**
- Eckstein, Jutta, 86, 92
 - ecosystem tuning, 46
 - enabling teams, 9, 86–90, 106–107
 - case study, 88–90
 - vs. communities of practice, 90

- defined, 187
- expected behaviors, 87–88
- engineering maturity, 73–74
- environment design, 50
- environmental scanning, 171
- Ericsson, 68
- Evans, Eric, 115
- evolution triggers, 162–164, 165–170
 - delivery cadence slowing down, 166–167
 - multiple business services relying on one large set of underlying services, 167–169
 - software too large for one team, 165–166
- expected behaviors
 - complicated-subsystems teams, 94
 - enabling teams, 87–88
 - stream-aligned teams, 85–86
- “Eyes On, Hands Off,” 46

F

- facilitating mode, 9, 134, 140–144
 - defined, 187
 - and reverse Conway maneuver, 147–148
 - team behaviors for, 143–144
- feature teams, 67–68
- The Five Dysfunctions of a Team: A Leadership Fable* (Lencioni), xx
- flow, enhancing, 148–151
- flow of change, 12, 63–64, 187
- focused communication, 24–26
- Forrester, Russ, xxii
- Forsgren, Nicole, 18, 64, 86
- four team types, 178–179
- Fowler, Martin, 115–116
- fracture planes, 115–123
 - business domain bounded context, 115–116
 - case study, 121–125
 - change cadence, 117
 - defined, 187
 - natural, 121
 - performance isolation, 119
 - regulatory compliance, 116–117
 - risk, 118–119
 - team location, 118
 - technology, 120
 - user personas, 120–121
- Fried, Jason, 55

G

- germane, 40
- Google, 70
- Google Cloud, 72
- group chat prefixes, 55–56
- Guide to Organisation Design* (Stanford), 38
- guilds, 49

H

- Hansson, David Heinemeir, 55
- Harvard Business School, 16
- healthcare organizations, 75–76
- Helfand, Heidi, 48, 118, 150

- Hering, Mirco, 120
- heuristics for domain assignment, 43
- hidden monoliths, 112–114
- high cohesion, 22
- high-trust organizations, 32
- homomorphic force, 10, 19–20
- Horizons, 36
- Hotchkiss, Dave, 157–159
- “How Do Committees Invent?” (Conway), 9–10
- Humble, Jez, 18, 36, 64, 86
- Humphrey, Andy, 53–55, 97–99

I

- IBM, 146–147
- IBM 8086 processor, 101
- Ikea, 47
- influences for book, xxi–xxii
- infrastructure automation, 11
- infrastructure teams, 105
- ING Netherlands, 53
- Ingles, Paul, 155
- interaction mode key, xx
- interaction modes, 179, 185
- intermittent collaboration, 133
- Internet of Things (IoT), 84, 101, 123–125, 156–157, 171
- inter-team communication, 25–26, 27
- Ivarsson, Anders, 50

J

- Java Virtual Machine, 101
- Jay, Graylin, 41
- joined-at-the-database monolith, 113, 188

K

- Kelly, Allan, 10, 24, 35, 101, 148
- Kim, Gene, 18, 64, 86, 172
- Kim, Kyung Hee, 136, 138
- Kniberg, Henrik, 25, 50, 100
- Knight, Pamela, xxii, 36
- Kotte, Gustaf Nilsson, 47
- Kotter, John, 161

L

- Lambert, Michael, 50–52
- “A Leader’s Framework for Decision Making” (Snowden and Boone), xxi
- Lean Enterprise* (Humble, Molesky, & O’Reilly), 36
- Lencioni, Patrick, xxii
- Lewis, James, xxii, 10, 34
- Linux, 101
- Lister, Timothy, 38
- log-aggregation tools, 27–28
- loose coupling, 22
- Luo, Jiao, 80

M

- MacCormack, Alan, 16
- Maibaum, Michael, 94–95, 162–164
- Make Space* (Doorley & Witthoft), 53

Malan, Ruth, xx, 17, 23, 148
 McChrystal, Stanley, 31, 46
 Microsoft, 69
 Millett, Scott, 116
 Minick, Eric, 146–147
 misplaced boundaries, 150–151
 missing capabilities, 150–151
 “A Model for Team-Based Organization
 Performance” (Forrester and Drexler), xx
The Modern Firm (Roberts), 23
 Molesky, Joanne, 36
 monolithic build, 188
 monolithic model, 114, 188
 monolithic rebuilds, 113
 monolithic release, 113, 188
 monolithic shared databases, 16
 monolithic thinking, 114, 188
 monolithic workplace, 114, 188
 monoliths, 112–114
 application, 113, 187
 hidden, 112–114
 joined-at-the-database, 113, 188
 multi-layer viable-systems model, 103
The Mythical Man-Month (Brooks), 35

N

Narayan, Sriram, 172, 173
 .Net Framework, 101
 Neumark, Peter, 92–93
The New Hacker's Dictionary (Raymond), 10
 new practices, adoption of, 155–159
 Nokia, 38–39
 non-blocking dependencies, 68–69
 Nygard, Michael, 22, 116

O

onion concept, 34
 open-plan office, 114
 Ops team, 80–81
 O'Reilly, Barry, 36
 org chart thinking, 3–14
 bottlenecks, 11–12
 cognitive load, 11–12
 collaboration mode, 9
 communication structures, 4–8
 complicated-subsystem teams, 9
 Conway's law, 9–11
 enabling teams, 9
 facilitating mode, 9
 platform teams, 9
 problems with, 5–7
 stream-aligned teams, 9
 team interaction modes, 9
 Team Topologies model, 9
 team types, 9
 thinking beyond, 7–8
 X-as-a-Service mode, 9
 organization design, 16–17, 23–24
 organization design evolution, 181
 organization size, 73–74

organizational sensing, 64–65, 153–175
 adoption of new practices, 155–159
 business as usual teams, 173–174
 business process management, 169
 case study, 154–155, 157–159, 162–164
 collaboration mode, 153–155, 159–161
 environmental scanning, 171
 evolution triggers, 165–170
 rapid learning, 155–159
 self-steering design and development, 170–174
 team topologies combination, 164–165
 team topologies evolution, 159–164, 165–170
 X-as-a-Service mode, 153–154, 161
Out of the Crisis (Deming), 38
 OutSystems, 11, 42

P

Pais, Manuel, 66
 Payment Card Industry Data Security Standard
 (PCI DSS), 117
 Pearce, Jo, 40
Peopleware (DeMarco & Lister), 38
 performance isolation, 119
The Phoenix Project, 166
 Pierce, Robert A., 136, 138
 Pink, Dan, 11
 Pivotal, 149–150
 Pivotal Cloud Foundry (PCF), 48–49, 101
 platform composition, 96–99
 platform teams, 9, 92–96, 105–106, 188
 platforms, 100–104
 cognitive load reduction, 101–102
 constraints, 102
 managed as a live product or service, 103–104
 multi-layer viable-systems model, 103
 product development acceleration, 101–102
 thinnest viable, 101, 184
 underlying, 102–103
 Poole, Marshall Scott, xix
 Poppulo, 121–123
 Prezi, 92–93
 principle of overlapping measurement, 143
The Principles of Product Development Flow
 (Reinertsen), 23, 143
 product teams, 68–69
 Project Myopia (Kelly), 35
 promise theory, 142

R

rapid learning, 155–159
 Rautert, Markus, 16
 Raymond, Eric, 10
 “Real Life Agile Scaling” (Kniberg), 25
 rebuild everything, 113
 Red Hat Open Innovation Labs, 53
 regulatory compliance, 116–117
 Reinertsen, Don, 23, 143
 relative domain complexity, 41–42
Remote: Office Not Required (Fried & Hansson), 55
 Rensin, Dave, 72

- reorganizations, 28
- responsibilities, splitting, 74
- responsibility restriction, 39–41
- risk, 118–119
- Roberts, John, 23
- Rother, Mike, 134
- Rubio, Andy, 50–52

S

- Schwartz, Mark, 4
- self-service capabilities, 69
- self-steering design and development, 170–174
- Sheehan, Stephanie, 121–123
- shuffling team members, 62
- silo reduction, 74
- single view of the world, 114
- site reliability engineering (SRE), 70–72
- Skelton, Matthew, 66
- Sky Betting & Gaming, 94–95, 162–164
- Snowden, Dave, xix
- software architecture as flows of change, 23
- software boundaries, 115–123
 - business domain bounded context, 115–116
 - case study, 121–125
 - change cadence, 117
 - defined, 187
 - natural, 121
 - performance isolation, 119
 - regulatory compliance, 116–117
 - risk, 118–119
 - team location, 118
 - technology, 120
 - user personas, 120–121
- software boundary size, 45–47
- software ownership, 36–37
- software scale, 73–74
- Sosa, Manuel, 24
- Spotify, 49, 50, 75
- squads, 50, 75
- standardization, 114
- Stanford, Naomi, 24, 38, 171
- static topologies, 61–78
 - ad hoc team design, 62
 - anti-patterns, 62
 - case study, 75–77
 - cloud teams, 69–70
 - compatible support systems, 69
 - credit reference agencies, 76–77
 - dependencies, 74–75
 - DevOps, 65–67
 - DevOps Topologies, 66–67, 75–77
 - engineering maturity, 73–74
 - feature teams, 67–68
 - flow of change, designing for, 63–64
 - healthcare organizations, 75–76
 - non-blocking dependencies, 68–69
 - organization size, 73–74
 - organizational sensing, 64–65
 - product teams, 68–69
 - self-service capabilities, 69

- shuffling team members, 62
- silo reduction, 74
- site reliability engineering, 70–72
- software scale, 73–74
- splitting responsibilities, 74
- team intercommunication, 64–65
- team patterns, successful, 67–72
- technical and cultural maturity, 72–73
- topology choice considerations, 72–75
- wait time between teams, 74–75
- stream-aligned teams, 9, 81–86, 104, 188
 - capabilities within, 83–84
 - case study, 82–83
 - expected behaviors, 85–86
- streams of change, suitable, 183–184
- support teams, 80–81, 107–109
- Sussna, Jeff, 85, 161, 172
- Sweller, John, 39–40

T

- team APIs, 47–56
 - benched bay approach, 51
 - case study, 50–52, 53–55
 - defined, 48, 188
 - environment design, 50
 - group chat prefixes, 55–56
 - guilds, 49
 - squads, 50
 - team interactions, 49
 - virtual environment design, 53–56
 - workspace design, 50–56
- team assignments, 22
- team behaviors, 141–144
- team habits, 134–135
- team interaction modes, 9, 131–152
 - awkward team interactions, 150–151
 - basic team organization, 146–148
 - case study, 146–147
 - choosing suitable, 143–145
 - collaboration mode, 133, 135–137, 142–143, 147–148, 149
 - effective APIs, 148
 - enhancing flow, 148–151
 - facilitating mode, 134, 140–144, 147–148
 - intermittent collaboration, 133
 - misplaced boundaries, 150–151
 - missing capabilities, 150–151
 - principle of overlapping measurement, 143
 - promise theory, 142
 - reducing uncertainty, 148–151
 - reverse Conway maneuver, 147–148
 - team behaviors for, 141–144
 - team habits, 134–135
 - temporary changes to, 149–150
 - well-defined team interactions, 132–133
 - X-as-a-Service mode, 133, 137–140, 143, 149
- team interactions, 49, 132–133
- team intercommunication, 64–65
- team lifespans, 35–36
- team location, 118

- Team of Teams* (McChrystal), 31, 46
- team patterns, successful, 67–72
 - cloud teams, 69–70
 - compatible support systems, 69
 - feature teams, 67–68
 - non-blocking dependencies, 68–69
 - product teams, 68–69
 - self-service capabilities, 69
 - site reliability engineering, 70–72
- team silos, 99–100
- team size, 32
- team topologies
 - capability gaps, 184–185
 - combining, 164–165
 - component teams to platform teams, 105–106
 - converting architecture and architects, 109
 - converting common to fundamental team topologies, 104–109
 - Conway's law, 15–29, 180–181
 - defined, 188
 - evolving, 159–170
 - four fundamental. *See* team types
 - four team types, 178–179
 - how to get started with, 183–185
 - infrastructure teams to platform teams, 105
 - interaction modes, sharing and practicing, 185
 - org chart thinking, 3–14
 - organization design evolution, 181
 - organizational sensing, 153–175
 - static, 61–78
 - stream-aligned teams, 104
 - suitable streams of change, 183–184
 - support teams, 107–109
 - team interaction modes, 131–152
 - team types, 79–110
 - team-first boundaries, 111–126
 - team-first thinking, 31–57, 179–180
 - thinnest viable platform, 184
 - three interaction modes, 179
 - tooling teams to enabling teams, 106–107
- team topologies combination, 164–165
- team topologies evolution, 159–170
 - adopting different interaction modes, 159–162
 - case study, 162–164
 - combining team topologies, 164–165
 - delivery cadence slowing down, 166–167
 - evolution triggers, 165–170
 - multiple business services relying on one large set of underlying services, 167–170
 - software too large for one team, 165–166
- Team Topologies model, 9
- team types, 9, 79–110
 - case study, 82–83, 88–90
 - complicated-subsystems teams, 91–92
 - converting common to fundamental team topologies, 104–109
 - enabling teams, 86–90
 - enabling teams *vs.* communities of practice, 90
 - four, 178–179
 - key, xx
 - Ops team, 80–81
 - platform composition, 96–99
 - platform teams, 92–96
 - platforms, 100–104
 - stream-aligned teams, 81–86
 - support team, 80–81
 - team silos, 99–100
- team-first boundaries, 111–126
 - application monolith, 113
 - business domain bounded context, 115–116
 - case study, 121–125
 - change cadence, 117
 - coupled releases, 113
 - distributed monolith, 112
 - fracture planes, 115–123
 - hidden monoliths, 112–114
 - joined-at-the-database monolith, 113
 - in manufacturing, 123–125
 - monolithic model, 114
 - monolithic rebuilds, 113
 - monolithic releases, 113
 - monolithic thinking, 114
 - monolithic workplace, 114
 - open-plan office, 114
 - performance isolation, 119
 - rebuild everything, 113
 - regulatory compliance, 116–117
 - risk, 118–119
 - single view of the world, 114
 - software boundaries, 115–123
 - standardization, 114
 - team location, 118
 - technology, 120
 - user personas, 120–121
- team-first mindset, 37–38
- team-first software architecture, 35
- team-first thinking, 31–57, 179–180
 - benched bay approach, 51
 - boundaries, 39–47
 - case study, 50–52, 53–55
 - cognitive load, 39–47
 - diversity, 38
 - domain identification, 43
 - domain limitations, 42–45
 - Dunbar's number, 32
 - ecosystem tuning, 46
 - environment design, 50
 - extraneous, 40
 - "Eyes On, Hands Off," 46
 - germane, 40
 - group chat prefixes, 55–56
 - guilds, 49
 - heuristics for domain assignment, 43
 - high-trust organizations, 32
 - horizons, 36
 - intrinsic, 40
 - onion concept, 34
 - relative domain complexity, 41–42
 - responsibility restriction, 39–41
 - small long-lived teams, 32–39

- software boundary size, 45–47
- software ownership, 36–37
- squads, 50
- team APIs, 47–56
- team definition, 32
- team interactions, 49
- team lifespans, 35–36
- team size, 32
- team-first mindset, 37–38
- team-first software architecture, 35
- trust and team size, 33–35
- Tuckman Teal Performance Model, 36
- types of, 39–40
- virtual environment design, 53–56
- workspace design, 50–56
- teams, small long-lived, 32–39
 - diversity, 38
 - Dunbar's number, 32
 - high-trust organizations, 32
 - horizons, 36
 - onion concept, 34
 - software ownership, 36–37
 - team definition, 32
 - team lifespans, 35–36
 - team size, 32
 - team-first mindset, 37–38
 - team-first software architecture, 35
 - trust and team size, 33–35
 - Tuckman Teal Performance Model, 36
- team-scoped flow, 21–23
- technical and cultural maturity, 72–73
- “Technical Consulting Teams,” 86
- technology for team-first boundaries, 120
- Thinking Environments*, 4
- thinnest viable platform (TVP), 101, 184, 188
- Thoughtworks, 10
- tool choice and communication, 27
- tooling teams, 106–107
- topology choice considerations, 72–75
 - dependencies, 74–75
 - engineering maturity, 73–74
 - organization size, 73–74
 - silos reduction, 74
 - software scale, 73–74
 - splitting responsibilities, 74
 - technical and cultural maturity, 72–73
 - topology choice considerations, 72–75
 - wait time between teams, 74–75
- Toyota, 134
- TransUnion, 76–77, 157–159
- Treyner, Ben, 70
- tribes, 75
- trust and team size, 33–35
- Tuckman, Bruce, xx
- Tuckman Teal Performance Model, 36
- Tune, Nick, 116

U

- uncertainty, reducing, 148–151
- unnecessary communication, 24–26

- user personas, 120–121
- uSwitch, 155

V

- version compatibility, 22
- virtual environment design, 53–56
- Vogels, Werner, 83

W

- wait time between teams, 74–75
- Watson, Ian, 76–77
- Weston, Robin, 88–90
- Whyte, Dave, 53–55, 97–99
- Wiener, Norbert, xix
- Wiley, Evan, 48–49, 149–150
- Windows, 101
- Witthoft, Scott, 53
- workspace design, 50–56

X

- X-as-a-Service mode, 9, 133, 137–140
 - and collaboration mode, 149, 153–154
 - defined, 188
 - and delivery predictability, 160–161
 - and evolution of team topologies, 159–161
 - team behaviors for, 143

ACKNOWLEDGMENTS

Writing a book is a collaborative task involving many people without whom the finished book wouldn't be impossible. We'd like to thank our peer reviewers for taking the time to give detailed feedback on the book: Charles T. Betz, Jeremy Brown, Joanne Molesky, Nick Tune, and Ruth Malan. We also want to thank the authors and originators of our case studies and industry examples: Albert Bertilsson, Anders Ivarsson, Andy Humphrey, Andy Rubio, Damien Daly, Dave Hotchkiss, Dave Whyte, Eric Minick, Fernando Cornago, Gustaf Nilsson Kotte, Henrik Kniberg, Ian Watson, Markus Rautert, Michael Lambert, Michael Maibaum, Miguel Antunes, Paul Ingles, Pulak Agrawal, Robin Weston, Stephanie Sheehan, and Wolfgang John.

We'd like to thank everyone who contributed to the original DevOps Topologies patterns, especially James Betteley, Jamie Buchanan, John Clapham, Kevin Hinde, and Matt Franz. A special thanks goes to John Cutler for a passionate outsider's view of the Team Topologies approach, and to Gareth Rushgrove for helping to expand the audience for the original DevOps Topologies patterns. We also want to thank our colleague Jovile Bartkeviciute at Conflux for her tireless research.

The team at IT Revolution Press has been amazing, especially Anna Noak, Lean Brown, and the other editors and designers—we've really valued their advice, support, and infectious enthusiasm. We're grateful to Gene Kim for inviting us to speak at DevOps Enterprise Summit in London in 2017, which helped us to realize the value of the emerging Team Topologies ideas.

Finally, we'd like to thank the people whose ideas, talks, and writing inspired us to become interested in the fascinating relationship between teams and software in the first place, and helped to make this book a reality: Allan Kelly, Andy Longshaw, Charles T. Betz, Donella Meadows, James Lewis, Gene Kim, Mel Conway, Mirco Hering, Rachel Laycock, Ruth Malan, and Randy Shoup.

ABOUT THE AUTHORS



Matthew Skelton has been building, deploying, and operating commercial software systems since 1998, and has worked for organizations including London Stock Exchange, GlaxoSmithKline, FT.com, LexisNexis, and the UK government. Head of Consulting at Conflux (confluxdigital.net), Matthew is the co-author of the books *Continuous Delivery with Windows and .NET* (2016) and *Team Guide to Software Operability* (2016). Matthew holds a BSc in computer science and cybernetics from the University of Reading, an MSc in neuroscience from the University of Oxford, and an MA in music from the Open

University; he is a chartered engineer (CEng) in the UK. In his free time, Matthew plays the trumpet, sings in choirs, writes music, and enjoys trail running.

Manuel Pais is an independent DevOps and Continuous Delivery Consultant focused on team design, practices, and flow. He helps organizations define and adopt DevOps and Continuous Delivery (both from technical and human perspectives) via strategic assessments, practical workshops, and coaching. Manuel is the co-author of *Team Guide to Software Releasability* (2018).

Matthew and **Manuel** have worked together on organization design for modern software systems with many clients around the world. Their training sessions on organization design for modern software systems have helped numerous organizations to rethink their approach to team intercommunication and software architecture, improving flow and the effectiveness of software delivery.

