Thinking Environments

Evaluating Organizational Models for DevOps to Accelerate Business and Empower Workers

DevOps Enterprise Forum 2016

Thinking Environments

Evaluating Organizational Models for DevOps to Accelerate Business and Empower Workers

DEVOPS ENTERPRISE FORUM
2016
IT REVOLUTION
PORTLAND, OREGON

25 NW 23rd Pl, Suite 6314
Portland, OR 97210

Thinking Environments: Evaluating Organizational Models for DevOps to Accelerate Business and Empower Workers
Copyright © 2016 IT Revolution

First Edition

Produced in the United States of America

10 9 8 7 6 5 4 3 2 1

Cover design and interior by Abbey Gaterud
Cover illustration by Mammoth

“Fix the environment, not the people.”

—David Marquet Captain, US Navy (Ret.)

Contents

PREFACE

In May of this year, IT Revolution once again had the pleasure to host 50 technology leaders and thinkers from across the DevOps Enterprise community at the DevOps Enterprise Forum in Portland, Oregon. The Forum's ongoing goal is to create written guidance, gathered from the best experts in these respective areas, for overcoming the top obstacles in the DevOps Enterprise community.

Gathering feedback and information from the 2015 DevOps Enterprise Summit, we narrowed down the four key areas identified by the community to tackle in this this years Forum papers:

- **Leading Change:** What are effective strategies and methods for leading change in large organizations?
- **Organization Design:** What do the organization charts look like for organizations successfully adopting DevOps? What are the respective roles and responsibilities, and how has it changed from more traditional IT organizations?
- **Modern Technology Practices:** What are modern architectural and technical practices that every technology leader needs to know about?
- **Compliance and Security:** What are concrete ways for DevOps to bridge the information security and compliance gap, to show auditors and regulators that effective controls exist to prevent, detect and correct problems?

For three days, we broke into groups based on each of the key areas and set to work, choosing teams, sometimes switching between teams, collaborating, sharing, arguing…and writing. After the Forum concluded, the groups spent the next six months working together to complete and refine the work they started together.

The end result can be found on the Forum page at IT Revolution web site (http://itrevolution.com/devops_enterprise_forum_guidance) and all the forum papers, from both this year and last year, are free to the community.

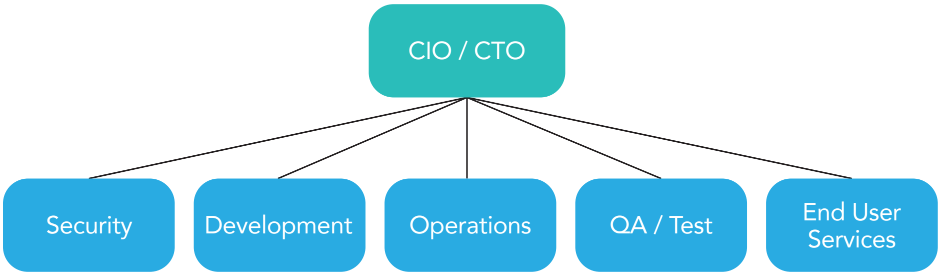
IT Revolution is proud to share the outcomes of the hard work, dedication, and collaboration of the amazing group of people from the 2016 DevOps Enterprise Forum, our hope is that you will gain valuable insight into DevOps as a practice.

Gene Kim
November 2016
Portland, Oregon

EXECUTIVE SUMMARY

While the traditional IT organization is structured into functional silos, DevOps relies on empowered, cross-functional teams. Is it possible to blend the two approaches and work within the traditional structure? Or do you need to restructure your organization to support DevOps? The traditional structure offered a way to continuously improve skills in individual practice areas—software development, infrastructure, operations, and security, for example. When you organize around cross-functional teams, do you lose the ability for more skillful infrastructure experts to oversee and coach more junior infrastructure specialists, and do you lose opportunity to build capability and expertise within that technical specialty? The traditional functional organization can be highly efficient at allocating expert across different projects, reassigning technical experts based on the company's needs. Must a DevOps organization sacrifice this efficiency?

These are questions that many of us grapple with as we travel along the DevOps transformation journey to improve our organizations. In this paper we address those questions, identify some of the models that enterprises and organizations are currently using, and propose some ideas that can help leaders as they plan their future.



INTRODUCTION —THE CHALLENGE

"To succeed consistently, good managers need to be skilled not just in choosing, training, and motivating the right people for the right job, but in choosing, building, and preparing the right organization for the job as well."

—Clayton M. Christensen, The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail

DevOps practices help organizations speed up development, innovate, and deliver high-quality products and solutions. But DevOps is not an organizational structure. Rather, it defines a way to organize independent teams (cross-functional or "full stack"), a culture (humane and outcome-based), a set of Lean principles (fast feedback, including feedback from production), and a set of practices (highly automated, continuous delivery). It remains for us to find the best way to fit those principles into an organizational structure for IT and for the enterprise as a whole.

There will not be a single solution that works for every organization; in fact, since DevOps is relatively new, we cannot be sure what works and what does not. Nor do we believe that organizational design is the only—or even the most important—factor in obtaining the value of the DevOps approach. As Mark Schwartz suggests in *The Art of Business Value*, each organization has, embedded in its corporate culture and in its rules and processes, its own understanding of business value and how to best create it.[1] Each organization will need to take its own experimental, Agile approach to finding which

1. Mark Schwartz, *The Art of Business Value* (Portland, OR: IT Revolution, 2016), chapters 3-4.

organizational structure works best. As with all other Agile approaches, W. Edwards Deming's plan-do-study-act (PDSA) loop[2] can be used to continuously refine the model.

Nevertheless, this white paper looks at models that are currently being used by organizations, the principles and assumptions behind each of the models, and the advantages and disadvantages of each in facilitating DevOps. We also discuss the goals of DevOps organization design: What are the characteristics an organization must have in order to support the use of DevOps to drive business value? What do we need to solve in setting up a DevOps friendly organizational structure? What trade-offs do we need to consider?

This paper is intended to help all those involved in driving a DevOps transformation. In particular, we hope it will be useful to:

- **CIOs and CTOs:** IT leaders exercise their influence, in part, through the organizational structures they put in place. The models and discussions in this paper will help CIOs and CTOs speed their transition to DevOps practices, convince their fellow CXOs of the virtues of their selected organization design, and get the most value from their DevOps transformation.

- **VPs and Directors of Engineering/IT:** This paper will help these IT leaders convince more senior leaders of the value and practicality of a transformation to DevOps, and provide ideas on how best to structure the organization to improve engineering performance and interactions between their specialties.

- **Line Leaders (managers of individual contributors):** These managers are the critical link in making the DevOps approach work tactically. They must find ways to encourage interactions between

2. W. Edwards Deming, "The PDSA Cycle," The W. Edwards Deming Institute website, November 28, 2012. https://www.deming.org/theman/theories/pdsacycle.

their teams, take advantage of the organizational structure to remove impediments for their employees, and translate the organization's vision into goals that their empowered teams can act upon.

WHY DOES ORGANIZATIONAL STRUCTURE MATTER?

In *The Leader's Handbook*, Peter R. Scholtes traces modern organizational design to October 1841, when two Western Railroad passenger trains collided in New York, killing a conductor and a passenger and injuring seventeen more.[3] In the aftermath, the need to establish formal accountability and blame led the company's directors to create the first organizational chart. With the advent of Fordism and Taylorism, companies saw the value of organizational design in improving efficiency through standardization and economies of scale. Modern approaches to organizational structure and purpose, best exemplified by Alfred Sloan's tenure as CEO of General Motors and his introduction of Management by Objectives (MBO) as a way to pass strategic objectives down through a hierarchy, may be understood as a refinement of these organizational patterns of the nineteenth century.

A very different view of the purpose and function of management emerged in the 1950s from the work of W. Edwards Deming and his insight that 94% of all failures can be traced to systemic issues, rather than problems with individual behaviors.[4] DevOps, grounded in Lean thinking like other Agile methods, may be understood as an extension of Deming's ideas to the contemporary enterprise and our post-industrial focus on economies of experience, where the transactional values that matter most extend beyond simple exchanges of currency to the relationships that are established between an enterprise and its customers.

3. Peter R. Scholtes, *The Leader's Handbook* (New York: McGraw-Hill, 1997), 2.

4. W. Edwards Deming, *The New Economics* (Cambridge, MA: MIT Press, 1994), 33.

In *Exponential Organizations*, Salim Ismail argues that even these twentieth-century advances in management thinking are grounded in a traditional mindset of managing a scarcity of resources and owning assets.[5] Such processes scale linearly and have traditionally led to an increasing depth of management hierarchy. Today, advancements such as the cloud and the Internet of Things are having a dramatic impact on the the way companies operate, allowing them to scale exponentially. Inevitably, this breaks the model of scaling organizational structures and is causing an urgent need to explore new organizational approaches.

In 1967, Melvin Conway proposed the principle that is now known as Conway's Law: "Organizations which design systems," according to Conway, "are constrained to produce designs which are copies of the communication structures of these organizations."[6] His law is generally interpreted to mean that organizational structure determines product design; or, alternatively, that designing the organization is a critical part of any product design decisions. It is also, we would add, a critical element of designing the processes which produce those products. The way that DevOps will be implemented depends crucially on the organizational structure in which it is implemented.

The Traditional Functional Silo Model

The traditional IT model, designed as a hierarchy based on functional silos, attempts to realize efficiencies by right-sizing the capacity of each silo to the "demand" from across the organization, and by sharing technical knowledge between the practitioners of a given specialty area. An IT organization is often divided, at its highest level, into separate groups for system development, infrastructure engineering, operations, and security, and often

5. Salim Ismail, *Exponential Organizations* (New York: Diversion Books, 2014).

6. Mel Conway, "Conway's Law," *MelConway.com*, accessed September 2016, http://www.melconway.com/Home/Conways_Law.html; "Conway's law," Wikipedia.com, last modified September 14, 2016, https://en.wikipedia.org/wiki/Conway%27s_law.

also includes specialized groups responsible for QA and testing. Each of these groups is organized as a hierarchy reporting to senior IT leadership. Because each group—let's take infrastructure engineering as an example—is organized independently, it can grow as the company's infrastructure needs grow, and shrink as they shrink; in other words, capacity can be optimized as the infrastructure engineering management tries to achieve full capacity utilization. Also, because the group is based around a functional specialization, it can transfer knowledge throughout its hierarchy, with more senior practitioners coaching those less experienced, and employees moving up the ranks as they gain knowledge and experience within their technical domains.

There are a few interesting things to note about this model. First, it fits well with the usual approach to budget accountability: the leader of the infrastructure engineering division can be given a budget which is then passed down the hierarchy; he or she can also be given responsibility for reducing the budget by finding cost efficiencies in the way the group handles infrastructure provisioning. Accountability is clear: if anything goes wrong with infrastructure, the company knows exactly whom to blame. The model also works well with strategic planning when strategy is viewed as "functional strategy,"[7] since the experts in each domain can formulate the strategy.

On the other hand, the functional silo organization is less able to manage information about business applications and users. Since each business initiative will require involvement from all of the functional silos, knowledge about the initiative and the needs it addresses will be dispersed among the silos. The silo organization will also be much slower to respond to business needs when those needs involve coordination between functions, since the silo organization is optimized for communication within the silo rather than between silos. Lean theory teaches us that there is waste any time there is a

7. Functional strategy in management theory is only one type of strategy, along with business strategy and corporate strategy. See, for example, Paul L. Drnevitch and David C. Croson. "Information Technology and Business-Level Strategy: Toward an Integrated Theoretical Perspective." *MIS Quarterly* 37 no. 2 (June 2013). 483-509.

"hand-off" between resources, and the functional silo organization guarantees that projects will need to be handed off a number of times. Finally, if we take an expansive view of Conway's Law, we might be concerned that our work will wind up structured by functional silos rather than by user-driven business needs.

DevOps, like other Agile and Lean approaches to IT delivery, relies on cross-functional delivery teams. It goes beyond other Agile and Lean approaches in the breadth it requires of its teams, which should have skills in development, testing, operations, and security, at the very least. Working in such cross-functional teams avoids hand-offs and increases the possibilities for rapid feedback—not only from testers and user reviewers but also from the actual usage of the system in production. There is clearly a tension between the DevOps cross-functional approach and the traditional functional silo based approach. Can they co-exist?

Characteristics of a DevOps Organization

Rather than asking whether DevOps can be made to fit within traditional organizational structures, it might be better to ask what characteristics an organizational structure would need to have in order to align with the DevOps model. We can then use these characteristics to evaluate proposed organizational structures to see how closely they fit. In effect, these characteristics are acceptance criteria for an organizational design, and companies can innovate and experiment with different models that might deliver on these requirements.

First, the organizational structure must support and promote the mechanics of the DevOps approach and its goals. As a type of Lean process, DevOps focuses on shortening lead times and generating rapid feedback. In evaluating an organizational structure, then, we must ask whether it is likely to deliver on these objectives. Are there barriers to rapid delivery? For example, are there difficult hand-offs between different groups? Are there required sign-offs or approvals that will result in "wait time?" Is the flow of feedback frictionless or does it need to go through "appropriate channels?" Can teams be autonomous and empowered, or would their goals conflict with those of the hierarchy they are part of? Does the organization "optimize the whole"—that is, does it manage the trade-offs of organizational structure design so as to get the best result on the whole?

Second, the organization must encourage communication and the free flow of information. In general, this means that communication should be face to face and ad-hoc, rather than through formal documentation. Silos tend to encourage communication within the silo but not between silos: Does the organization's hierarchy encourage communication in the right "direction" to accomplish the goals of DevOps?

Third, the organizational structure must coordinate accountabilities to support the goals of delivering high-quality, impactful software. DevOps requires empowered teams, which in turn requires that the teams be given responsibility of a goal. Does the organizational structure allow meaningful goals to be passed down through the structure so that teams can "own" their delivery?

Fourth, are the needs for risk mitigation, compliance, and auditability. Does the organizational structure establish the controls required by the company's compliance regimes (SOX, HIPAA, FISMA, etc.) while at the same time providing the flexibility and empowerment for DevOps teams to succeed? Does the IT organization's structure fit logically into that of the rest of the company? Is it likely to produce the results required by the enterprise as a whole?

Fifth, the organization must allow for a humane and fair distribution of burdens. DevOps is based on the idea that a development team should not "toss its product over the wall" to an operations team which then must deal with the operational consequences, nor should an operations team so constrain development that it is unable to meet the goals of its users. This principle should apply to the organization as a whole—are responsibilities distributed such that no group has an unsustainable burden? This is really a way of asking whether the entire organization is motivated by a shared goal rather than by conflicting goals.

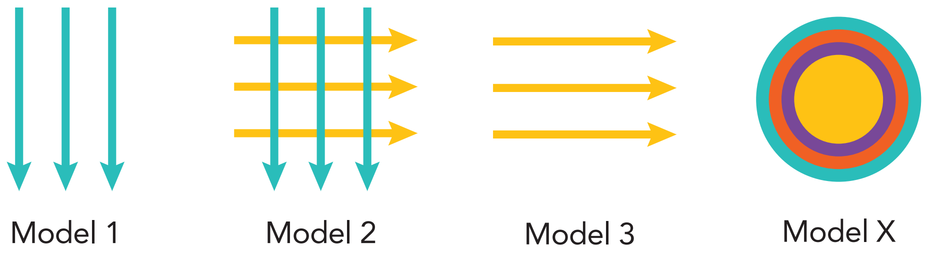
Sixth, the organizational structure should recognize the value and the voice of individuals. Just as individuals in the Toyota manufacturing process are encouraged to pull the *Andon* cord if they see a problem, DevOps and other Lean practices seek to empower those closest to the action to use their own judgment. Does the structure require that individuals get layers of approval to take action? Or, is flattened in a way that encourages interaction between managers and individual contributors? Do individuals have a voice in decisions? Is their feedback regularly solicited?

Seventh and finally, the organization must be flexible enough to support continuous improvement. Is it self-organizing in the appropriate degree? Can it easily reconfigure itself based on learnings and shared experience?

The structure of the organization is not the only factor that encourages these behaviors, but it certainly can be an obstacle. With these characteristics in hand we can examine some organizational structures and the degree to which they deliver them.

ORGANIZATIONAL MODELS

To simplify the landscape and form a useful basis for discussion, we have identified four representative models, based on our discussions with a number of large organizations that have implemented DevOps approaches. For each model, we identify its advantages and disadvantages for supporting DevOps, and we provide suggestions on how to make the most of the model. We also identify steps for transitioning from the model to higher maturity structures.



Model 1 is focused on vertical organization, Model 2 is focused on vertical organization and adds a horizontal component, Model 3 is focused on primarily horizontal organization, and Model X is focused on the holistic organization in multiple dimensions at once. Model X is the most different of the 4, as it removes all elements of management hierarchy.

The first three models, we believe, are widely distributed. The fourth, which we've named "Model X," is a much rarer species of organization and more than anything represents a glimpse into a future-state model that—looking at how organizations today are typically structured as a whole—lies "beyond" contemporary organizational practices and thinking.

- Model 2: Matrix—functional silos with additional formal or informal lines of reporting to a cross-functional product/service or project team.
- Model 3: Product—"full stack," cross-functional organization organized around products or portfolios.
- Model X: Adaptive—organic and dynamic structure that adjusts and reconfigures itself—the newest type of organizational structure.

As companies have adopted DevOps practices, there seems to be a tendency to evolve in a gradual fashion from one model to the next, making use of next level structures to stimulate greater maturity in DevOps practices and agility. Conceptually, at least, this incremental approach makes sense considering DevOps' emphasis on experimentation and continuous improvement. This is not to say that a decision to drive a more dramatic transformation is incorrect in all circumstances; a change in structure such as moving from Model 1 straight to Model 3 (effectively bypassing Model 2) could in some contexts help achieve a profound alignment to DevOps principles and practices. At the end of the day, however, there is no default answer to the question: "What model is best for my organization?" In this way, we do not offer "best practices" for organizational design. Invariably, it depends.

A decision to restructure an organization should never be made lightly, and typically requires thinking strategically through numerous factors. Of those factors that must be evaluated when making a decision to reorganize, consideration of organizational culture is particularly important. As MIT Professor Emeritus Edgar Schein has written, "Culture determines and limits strategy."[8] In this context, it can be helpful to consider the organizational models presented here from the perspective of the work of pioneering thinker Frederic

8. Edgar Schein, *Organizational Culture and Leadership*, (San Francisco, CA: Jossey-Bass, 1985), 377.

Laloux, author of *Reinventing Organizations*.[9] Laloux describes a typology of organizational culture within an evolutionary framework, where transition from one type of organization to another represents an underlying transition to a new stage in human consciousness. The table on the next page demonstrates how the four models we have presented in this paper correlate with Laloux's model of evolutionary development.

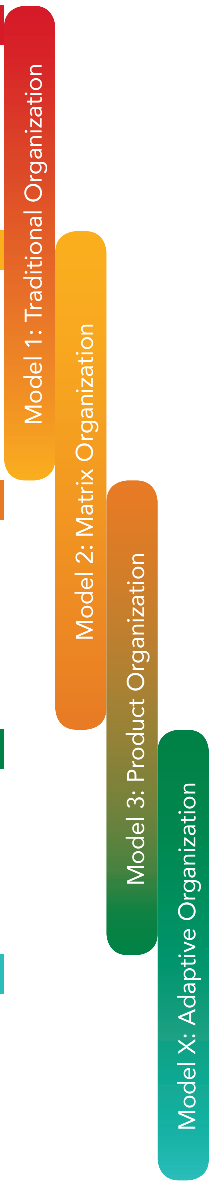
The structure of an organization may not always correlate 100% with its culture or guiding principles. An organization that has evolved its culture over time may contain structural elements that are no longer particularly relevant from the perspective of how work is organized, much like vestigial organs that may have once performed an important function earlier in the evolutionary history of a species. Nevertheless, this comparison of organizational structure to organizational culture does provide a useful way to think about how a transition to a new structure can be guided by assessment of current and desired culture.

This paper does not pretend to represent every enterprise or organization, but we believe the four models described below can be used to assess the current state within an organization and be a guide for leaders when considering future designs. Changing an organization's structural model will not magically transform its business or its culture, but using the information from the models below may help leaders at inflection points in their DevOps transformational journey.

9. Frederic Laloux, *Reinventing Organizations*, (Brussels: Nelson Parker, 2014).

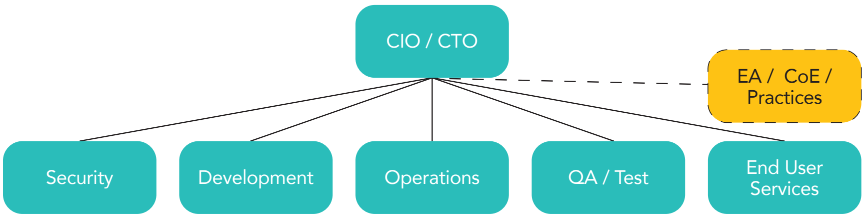
Exhibit 1: Evolutionary Breakthroughs in Human Collaboration

COLOR	DESCRIPTION	GUIDING METAPHOR	KEY BREAKTHROUGHS	CURRENT EXAMPLES
RED	Constant exercise of power by chief to keep foot soldiers in line. Highly reactive, short-term focus. Thrives in chaotic environments.	Wolf pack	• Division of labor • Command authority	• Organized crime • Street gangs • Tribal militias
AMBER	Highly formal roles within a hierarchical pyramid. Top-down command and control. Future is repetition of the past.	Army	• Formal roles (stable and scalable hierarchies) • Stable, replicable processes (long-term perspectives)	• Catholic Church • Military • Most government organizations (public school systems, police departments)
ORANGE	Goal is to beat competition, achieve profit and growth. Management by objectives (command and control over what, freedom over how)	Machine	• Innovation • Accountability • Meritocracy	• Multinational companies • Investment banks • Charter schools
GREEN	Focus on culture and empowerment to boost employee motivation. Stakeholders replace shareholders as primary purpose	Family	• Empowerment • Egalitarian management • Stakeholder model	Businesses known for idealistic practices (Ben & Jerry's, Southwest Airlines, Starbucks, Zappos)
TEAL	Self-management replaces hierarchical pyramid. Organizations are seen as living entities, oriented toward realizing their potential.	Living organism	• Self-management • Wholeness • Evolutionary purpose	A few pioneering organizations


Model 1: Traditional Organization
Model 2: Matrix Organization
Model 3: Product Organization
Model X: Adaptive Organization

Source: Frederic Laloux, Reinventing Organizations (Nelson Parker, 2014)

MODEL 1—TRADITIONAL FUNCTIONAL SILO MODEL



As we described in the introduction, IT organizations have traditionally been organized into functional silos—that is, independent hierarchies for each technical specialty. Though the names often differ and functions are sometimes combined, there are usually separate hierarchies for application development, infrastructure engineering, operations, security, quality assurance, helpdesk and user support, and perhaps enterprise architecture. While this model benefits from leveraging economies of scale to drive efficiencies and cost savings, it does create a very siloed organization that often results in many of the problems DevOps is meant to help address. Sometimes there is a Project Management Office (PMO) that tries to overcome this siloing by providing cross-cutting oversight of projects.

This model is a natural fit with traditional command and control management patterns; it is not an obvious fit with team-based approaches like that of DevOps. In large, traditional organizations, decision-making was clustered at the "top" of the organization—necessitating better visibility up the chain, and helping those at the top make informed decisions. Team-based organizations, on the other hand, migrate decision authority to empowered, decentralized teams, allowing decisions to be made by those "closest to the action," without the need to transfer information and implementation decisions up and down a management chain.

The traditional model has the weaknesses of any large, centrally controlled organization—it slows down change, since it tends to produce a web of dependencies, with each silo having to wait for others to provide information or approvals, and change must wait for information transfer and hand-offs up and down the management structure. Such organizations tend to be static and unambiguous in how information flows, inhibiting the feedback and learning required to promote continuous improvement and mature DevOps practices. Furthermore, applying Conway's Law (and validated by empirical research[10]), the traditional model results in products with tightly coupled, inflexible architectures. While applying DevOps in the context of this model may help overcome some of its weaknesses, it is clear that of all the models under consideration, this one provides the least suitable environment within which to mature DevOps practices.

KEY CHARACTERISTICS/BENEFITS/DRAWBACKS

ACCOUNTABILITY

Key Characteristic: Clear single points of accountability for the quality of the individual functions, but accountability for cross-cutting concerns is only at the top (CIO) level.

Benefits (Pros): By combining people in the same functional specialty into a single organizational silo, this structure allows specialists to learn together, define standards, and train and mentor new hires in the area of specialty. It encourages consistency of execution of the function, thereby leading to cost efficiencies. It is a structure that makes it easy to establish culpability for failure and to reward success.

Drawbacks (Cons): The traditional functional structure promotes "local"

10. Alan MacCormack, John Rusnak, Carliss Baldwin, "Exploring the Duality between Product and Organizational Architectures: A Test of the "Mirroring" Hypothesis," *Research Policy* 41, no. 8, (October 2012): 1309–1324.

optimization at the expense of end-to-end optimization and effective delivery of outcomes. Each sub-hierarchy within the overall structure is focused on finding efficiencies in its own operations, rather than on improving overall value delivery (which requires cross-silo optimization). In fact, accountability for IT's impact on business performance may be unclear below the level of the CIO. For example, if the development function created a new feature that was released to production and caused a security incident resulting in a service outage that caused the business to lose revenue, who was accountable—development, testing, operations, or security? Individual functions do not have enough control to deliver outcomes; for example, a testing team alone will not be capable of improving quality despite its ability to measure it.

BUDGETING

Key Characteristic: The traditional model allocates budget dollars based on functional needs rather than on outcomes (often capital investment projects use an orthogonal process, allocating funds for cross-functional outcomes). IT is thereby treated as a cost, not a value creator (part of the secondary value stream). The budgeted funds are divided and passed down through the organizational hierarchy.

Benefits (Pros): By cutting budgets, the organization can encourage functional areas to find cost efficiencies. This can reduce total IT spending to free up capital for strategic initiatives. The clear budget accountability focuses functional units on improving margins through reducing operating expenses.

Drawbacks (Cons): In an increasingly digital economy, the "business" of IT is core to the business itself. Cost reduction may lead to opportunity costs, thereby reducing overall business value across the enterprise. Reducing operating expenses is subject to diminishing returns in comparison to increasing revenue, which may have unbounded possibilities. The functional organization obscures these trade-offs, making it difficult to relate individual costs to their impact on the business as a whole.

FLOW AND QUEUEING

Key Characteristic: Hand-offs are required as work makes its way from one functional silo to another.

Benefits (Pros): This system encourages high capacity utilization: the size or capacity of each functional silo can be set based on its "demand." It also allows for a kind of check and balance or separation of duties, as each silo is able to apply its skills and knowledge to the work item when the item moves into its workstream.

Drawbacks (Cons): Each hand-off between silos creates process waste and increases lead time for the work item. Overhead is also added as each silo must communicate what is required of the next silo. Variances in demand cause queues to build up for different silos, which also increases work in process and lead time. It becomes more difficult to understand the status of a work item or project its delivery time, since it is subject to competing priorities as it enters each silo workstream.

PEOPLE

Key Characteristic: People in this model are treated as resources—"capacity" to complete work within each stage of a process.

Benefits (Pros): Because this model does not depend on close, frequent contact between team members, it lends itself to geographically dispersed workforces, hiring people or contractors in areas where they are least expensive.

Drawbacks (Cons): Treating people as "resources" or "capacity" does not make full use of their abilities; the additional power of a diverse team working together on a problem is considerable. People are more motivated when they have influence on outcomes, rather than just their own outputs. The traditional model is based on a factory metaphor, where the people are performing functions that would better be performed by machines, if the machines were capable. Knowledge work is very different: to be successful, one has to engage employees more fully.

PORTFOLIO MANAGEMENT

Key Characteristic: Work is organized by management and passed down through the hierarchical structure.

Benefits (Pros): Work can be easily and directly linked to strategy. Management has visibility into the work as it is delivered and can easily see how it is accomplishing strategic objectives.

Drawbacks (Cons): This approach disempowers employees, giving them the message that they don't need to think for themselves. It works best with well-structured problems with known, documented solutions, but is less effective with new and poorly understood problems. It can hinder the flow of work that occurs orthogonally to the structure of the organization, and makes it hard to pivot to meet changing needs.

STRATEGY LINKAGE

Key Characteristic: This model pushes strategy down through the management hierarchy, often through Management by Objectives.

Benefits (Pros): Through clear responsibilities for strategy setting and implementation, it promotes control and alignment of strategic decisions.

Drawbacks (Cons): One-way communication of strategic decisions prevents feedback to improve those decisions as they are formulated and executed. Unless strategy is very well communicated at all levels of the organization, lower level employees may not understand the reasoning behind strategic decisions that impact their work, making strategy execution more difficult to achieve.

SCENARIO: A MAJOR PRODUCTION INCIDENT REQUIRING A CODE FIX.

Key Characteristic: Most likely there will be a dedicated team for handling production incidents. If so, they will take the fastest route for addressing the incident. Their changes will then feed into a much slower process for merging and integrating with the next major release.

Benefits (Pros): No organizational change is typically required. Because the team is dedicated, development teams aren't constantly interrupted by production issues.

Drawbacks (Cons): There is limited feedback to development teams, reducing the chances for continuous improvement and learning. Without this feedback, it is likely technical debt will pile up, resulting in increased pressure on development teams to help operations fix recurring problems. This problem is made worse if—as is sometimes the case—the teams dedicated to fixing production issues are composed of inexperienced (often outsourced) staff with limited understanding of the systems.

EVALUATION

🟩 = Promotes DevOps adoption
🟨 = Minor area of risk for DevOps adoption
🟥 = Major area of risk for DevOps adoption

🟥	Promotes DevOps, short lead times
🟥	Encourages collaboration and communication (silos defeat communication)
🟥	Right kind of accountability (get on the right queue), routing instructions
🟥	Flat organization
🟥	Optimize the whole—deal with overall strategy, standards, etc.
🟥	Self-organizing overall structure—Can it reconfigure itself easily?

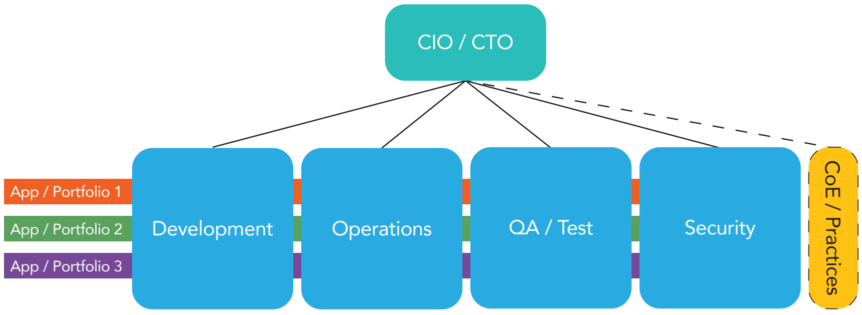
- Implements any necessary risk mitigation and compliance
- Must fit into the rest of the organization

Ideas to Help Transition to a "Better" Model

Organizations attempting to roll out DevOps within a Model 1 organizational structure will face a number of difficulties. As these difficulties accumulate, it might make sense to transition to one of the other models. The most glaring problem with Model 1 is that it requires multiple hand-offs between functional silos to complete a single work item; a second major problem is that no one is explicitly accountable for an entire work item, since it crosses the bounds of different management hierarchies. On the other hand, Model 1 allows each functional area to become more masterful in its specialty. One way organizations try to solve these two issues while retaining the strengths of Model 1 is by adding a "matrix" reporting structure on top of the Model 1 functional structure. With the matrix, teams can be assigned to outcomes while still retaining their functional reporting structure.

MODEL 2—MATRIX MODEL

The matrix model (in all its variations) is similar to the Traditional Functional Silo Model (Model 1), with additional layers of structure. Although individuals formally report to their functional silos, they are also assigned (formally or informally) to specific product portfolios. These product portfolios typically comprise one or more applications, although some may be more infrastructure focused. Team members therefore have a dual reporting responsibility: to their functions and to their products.



Model 2a: Shared Services Matrix Model

Such a model appears to address the most critical weaknesses of Model 1: in the Matrix model, individuals can feel a sense of ownership over a business outcome—their application or portfolio—while at the same time advancing their skills and gaining respect within their functional specialty.

The overhead of hand-offs between functional silos is reduced because the hand-off is just between individuals assigned to the particular product, so the work item is not waiting in a general queue for the functional silo. On the other hand, this model adds some overhead to the product portfolio team as the separate discipline leaders pull their team/experts together to best facilitate knowledge transfer and cross-portfolio collaboration opportunities. It also can be challenged by a scarcity of experts— limited number of functional experts means that product portfolios are competing for these individuals, which has the potential of delaying projects.

The problem, of course, is the ambiguity of the dual reporting structure, and is illustrated by an anecdote from the TV show *Game of Thrones*. In one of the show's episodes,[11] a riddle is presented: the teller asks what would happen if a king and a rich man each simultaneously instruct a bounty hunter to kill the other. Who has more power—the king or the rich man? The answer is that it is neither the king nor the rich man who holds the power in this situation, but the bounty hunter who is now placed in an awkward and perhaps philosophical position of deciding which has the most merit. Translating this to the Matrix Model where an engineer has a dual reporting structure, as long as the two bosses agree, direction is clear and progress is made. When they disagree, the engineer may decide based on merit. Or in a worse case scenario, the conflict may be completely debilitating to the individual and the team.

The Matrix Model, in a way, is not a perfect solution to the organizational structure challenge, as this model is inherently ambiguous. Its result will depend on the contexts and individuals involved. The individuals may align themselves more closely with their functions or with their projects. The advantages and disadvantages of the matrix will, therefore, depend on the direction of alignment, which may vary individual to individual.

At a high level, we can say that:

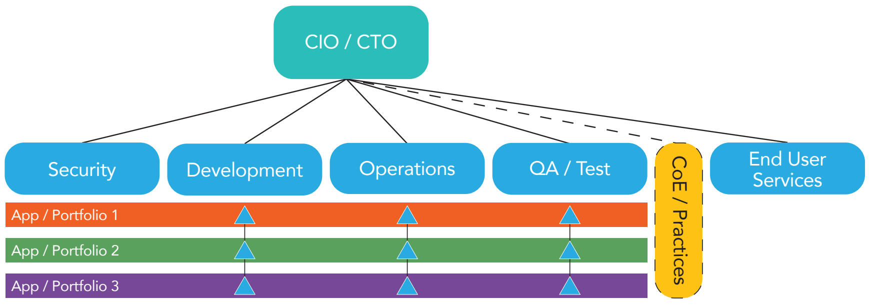
11. George R.R. Martin, David Benioff, D.B. Weiss, and Bryan Cogman,"What is Dead May Never Die," *Game of Thrones*, season 2, episode 3, directed by Alik Sakharov, aired April 15, 2012, HBO.

- If the alignment is too strong toward **function**, then product delivery may suffer at the expense of building efficient and predictable processes.

- If the alignment is too strong toward **project**, it might be to the detriment of the function and the product after the project is over. For example, a system may end up with all the requested functionality but not be operable or secure.

- If the alignment is too strong toward **product**, then looking at the organization as a whole, there might be an over-proliferation of technologies, tools, and methodologies which unnecessarily destroy economies of scale and the mobility of people within the organization.

A variation of this model, the Embedded Functions Matrix Model (see graphic below), aligns the teams more strongly with products and portfolios, and loosens some of the ties to the Traditional Functional Silo Model.



Model 2b—Embedded Functions Matrix Model

The Embedded Functions Matrix Model(2b) is similar to a traditional Shared Services Matrix Model (2a) due to the dual reporting relationships that it creates. However, unlike a traditional matrix, the individuals in this variation are completely and indefinitely embedded within the product teams. They will maintain a loose association with other individuals in other product teams via their reporting structure and various forums arranged by those managers for cross-team collaboration, but they are typically requested by and fully funded by the product team in which they are embedded, thereby taking on the identity as a committed member of their particular product group. While maintaining established reporting structures and continuing to encourage cross-product team information sharing, practices, and skills development, this structure also reduces communication latency and provides a shared sense of mission and accomplishment. The danger, as listed above, is that in this product focus there might be an over-proliferation of technologies, tools, and methodologies in the larger organization.

In some organizations, a "practices" team provides global training on skills and practices, and promotes community among technologists. This group will often participate with hands-on work in product teams on an ad-hoc basis to help disseminate best practices and tools (e.g., moving to or enhancing CI/CD pipelines or cloud hosting platforms). This team can also help mitigate some of the negative effects of a hyper product focus across the larger organization, by providing common technologies, tool, and methods.

KEY CHARACTERISTICS/BENEFITS/DRAWBACKS

ACCOUNTABILITY

Key Characteristic: These models use a matrixed organization style, where accountability is owned by more than one reporting group. It maintains an alignment of people to functional groups, but also maps them into portfolio families composed of applications or products. Funding may come from either the portfolio (Model 2b) or the functional group (Model 2a). When funded by the portfolio group, a natural free market effect may be created that prevents the functional groups from over-optimising as they might do in a Model 1. If the functional group does not effectively serve its portfolios, it will see demands on its services drop and portfolios turn to other providers (for example, hiring their own functional specialists directly). In this way, the dynamic forces the reduction of friction and latency between the functional roles.

Benefits (Pros): The matrix model could promote a healthy dialogue between functional disciplines and product portfolio leaders to align objectives and drive collaboration across portfolios.

Drawbacks (Cons): Individuals can receive conflicting direction from their 2+ bosses. This, of course, can be mitigated by designating one of the other supervisors as the "direct" reporting relationship, and the other(s) as the "dotted line."

BUDGETING

Key Characteristic: These matrix models allow the company discretion in awarding funding through either the functional line or the product portfolio. This allows for a clearer link between spending and the strategic—or at least product—objectives it is intended to support.

Benefits (Pros): In Model 2a, product portfolios are able to use functional experts without having to directly manage their costs. In Model 2b, the embedded teams are funded directly by the product portfolios, resulting in a tighter alignment with product needs and related expenses. Market driven expansion or flexibility of the product could dictate a demand to scale up the team, placing the control and costs immediately within the hands of the product owners rather than the more distant functional leaders. The true cost of a product or project is more apparent.

Drawbacks (Cons): For Model 2a, product portfolios are at the mercy of functional leaders in the growth and expansion of the number of these experts. The true cost of a product or project is only derived indirectly. In Model 2b, on the other hand, costs for functional team members are directly allocated and funded by the product portfolio teams.

FLOW AND QUEUEING

Key Characteristic: Functional teams are incentivized and motivated to increase their responsiveness because of their alignment with product teams; in Model 2b, this incentive is even stronger as the funding comes from those product portfolios.

Benefits (Pros): The functional teams are motivated to reduce queueing and optimise response time. This may be formalised as an operator level agreement from the functional team to the paying product team.

Drawbacks (Cons): Will likely require queues, hence delays, and management overhead to prioritise.

PEOPLE

Key Characteristic: As the with the Traditional Functional Silo Model, professional development is aligned with the function or area of expertise. At the same time, the embedded experts can focus on the product they are associated with.

Benefits (Pros): The concepts behind the matrixed model are well understood in the industry. This model can be easier to implement where there is no political motivation to completely deconstruct the traditional IT organization (this model could, therefore, be considered a transitional step in moving from Model 1 to Model 3). Individuals are able to develop their careers and skills in concert with a leader who focuses on their technical aptitude and professional expertise. Development, training, and performance reviews can be aligned with the functional area. Individual experts are deeply involved with the product portfolio, promoting trust, understanding, and speed.

Drawbacks (Cons): Depending on the degree of affinity with the product portfolio, technical professional development expectations might not be set or measured, or expectations for development outside the functional area might not be met. People may experience a tension between the requirements of function and product. Embedded experts can become myopic, reinvent solutions developed in other portfolios, and miss opportunities to share best practices and innovation with other areas. Individuals can also become stale and miss out on functional evolution and enhancements. This can be mitigated by establishing a practices function or guild organization responsible for promoting training, ongoing development, and good practices.

PORTFOLIO MANAGEMENT

Key Characteristic: Work is organized by the product leaders and distributed to the cross-functional product teams.

Benefits (Pros): Product leaders are empowered to move quickly with the support of teams that include full stack functional expertise and can thereby execute and deliver with agility.

Drawbacks (Cons): Opportunities for synergy between products can be missed, or duplication can occur between different product teams.

STRATEGY LINKAGE

Key Characteristic: Like the Traditional Functional Silo Model, matrix organizations transfer strategic objectives from the top down.

Benefits (Pros): Through clear responsibilities for strategy-setting and implementation, this model promotes control and alignment of strategic decisions.

Drawbacks (Cons): One-way communication of strategic decisions prevents feedback to improve those decisions as they are formulated and executed. Unless strategy is very well communicated at all levels of the organization, lower evel employees may not understand the reasoning behind strategic decisions that impact their work, making strategy execution more difficult to achieve.

SCENARIO: A MAJOR PRODUCTION INCIDENT REQUIRING A CODE FIX.

Key Characteristic: Each function has a dedicated team to address production incidents for their product.

Benefits (Pros): Because the functional teams are dedicated, the combined product group isn't constantly interrupted by issues involving other functional/product areas. The tighter integration of the functions can help provide greater insight into production problems for improvement (reinforcing feedback loop). The group can more easily combine and prioritize improvements in the same backlog and sprint cycles as feature development . For some tightly integrated embedded teams, development teams may even be included as part of the on-call rotation.

Drawbacks (Cons): Because the reporting structure remains the same as Model 1, there can be a temptation for the functional teams to maintain their silos, despite being part of a combined team, resulting in issues and feedback being hidden, reducing the chances for continuous improvement and learning.

EVALUATION

🟩	= Promotes DevOps adoption
🟨	= Minor area of risk for DevOps adoption
🟥	= Major area of risk for DevOps adoption

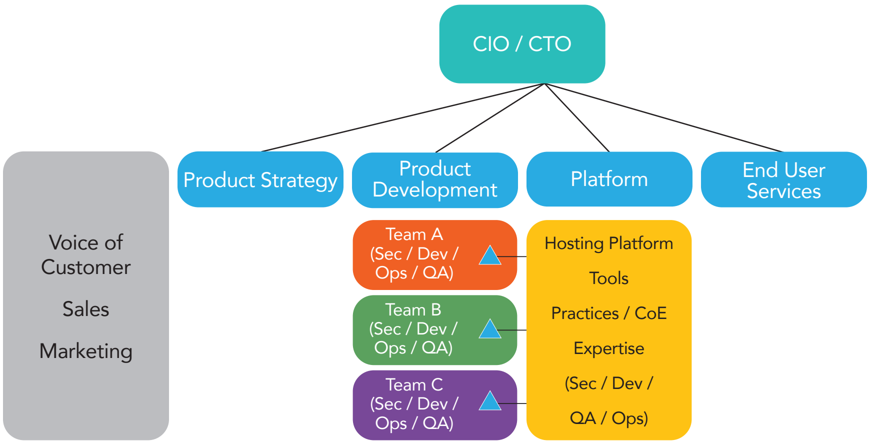
🟩	Promotes DevOps, short lead times
🟩	Encourages collaboration and communication (silos defeat communication)

🟨	Right kind of accountability (get on the right queue), routing instructions
🟨	Flat organization
🟨	Optimize the whole—deal with overall strategy, standards, etc.
🟨	Self-organizing overall structure—can it reconfigure itself easily?
🟩	Implements any necessary risk mitigation and compliance
🟩	Must fit into the rest of the organization

Ideas to Help Transition to a "Better" Model

Model 2 (especially 2b, Embedded Functions Matrix variant) is a great option for larger organization with strong centralized teams that want to begin to realize the benefits of tighter Dev and Ops integration without facing the political battles (and possible temporary organization chaos) needed to completely realign the organization into combined full-stack product teams. The transition from Model 1 to Model 2 has happened successfully for several organizations and has resulted in the expected improvements in cycle time, collaboration and quality. Organizations that want to continue to evolve to a more solidified full-stack product organization could eventually transition to Model 3 (described below) or incrementally pivot from Model 2 to Model 3 on a product-by-product basis. Using the staged approach, organizations can experiment and measure to land on a structure that is optimized for their business and teams.

MODEL 3—PRODUCT AND PLATFORM MODEL



The Product and Platform Model is based around full-stack, cross-functional teams dedicated to individual products or product groups. They are supported or enabled by a group that maintains the platform on which they work, and often additional groups for tools support, helpdesk, and end-user services. Their activities are informed by groups dedicated to the voice of the customer and product strategy.

We found that the exact functions included in the platform group varied from organization to organization, but tended to include the functions that help the product teams work most effectively. The platform group acquires or creates the infrastructure platform on which the full-stack teams develop, deploy, and operate their systems. They often provide and configure the tools that the product teams use—the continuous integration and deployment tools, development and testing platforms, and monitoring tools, for example. Quality assurance can be located among the platform groups, as well as performance experts such as site reliability engineers (SMEs).

Interestingly, a number of organizations also include functions in the platform group that are intended to help the product teams adopt best practices and refine and improve their crafts. In some cases, this includes centers of excellence in various specialties; agile coaching; a "dojo" that developers periodically join to learn or freshen their crafts; and experts in technical disciplines that can be lent to product teams when special expertise is needed. The platform group—in effect—allows the teams to be autonomous and productive, as external dependencies are provided and cared for.

The development teams themselves are fully aligned with the products they support. Each team is a full-stack DevOps team with responsibility for design, development, testing, infrastructure provisioning and configuration, deployment, operations, and security for their product. They are able to use the platforms and tools provided by the platforms group, and to solicit whatever other support they need from the experts and centers of excellence provided to them.

KEY CHARACTERISTICS/BENEFITS/DRAWBACKS

ACCOUNTABILITY

Key Characteristic: Teams are formed around products as defined by the perspective of the business.

Benefits (Pros): This organization structure promotes self-organization of teams and gives them end-to-end accountability for service delivery.

Drawbacks (Cons): Because development teams are autonomous and self-organizing, it may lead to proliferation of technologies, purchased software, processes, standards, and skill sets—thereby missing out on economies of scale. This, of course, is mitigated to the extent that common tools are provided by the platforms team. It may lead to duplication and internal conflict between overlapping products. Friction may develop between what is shared and what is owned by the individual product teams.

BUDGETING

Key Characteristic: DevOps teams may be funded by the product line they support, while the common platform can be funded separately.

Benefits (Pros): Because the product team oversees the budget, they may feel more free to find innovative ways to deliver value, or may be able to move more nimbly to implement features that add value. This is different from Models 1 and 2, where budgeting is tied more to functions—generally more concerned with cost savings and "operational efficiency"—rather than to product groups focused on maximizing value.

Drawbacks (Cons): There is a corresponding risk that operational efficiencies may be overlooked, especially those that could result from synergies between different products or common functional approaches between teams. The risk is that operational work is deprioritized to focus more on delivering "new business value." If model requires co-location, it might add more cost.

FLOW AND QUEUEING

Key Characteristic: Work is accomplished with far fewer hand-offs than in the traditional silo model.

Benefits (Pros): The focus is on the product and minimising waiting time. Speed and agility are enabled.

Drawbacks (Cons): Formal processes may not emerge, making it harder to predict time requirements and maintain quality standards.

PEOPLE

Key Characteristic: The people who thrive in this environment are full-stack, cross-functional team players, or T-shaped people.

Benefits (Pros): This structure encourages collaboration and teamwork to solve problems, taking people out of their comfort zones to grow, experiment, and learn new skills. The alignment to business values creates a higher likelihood that work will be meaningful.

Drawbacks (Cons): Without functional leaders, some types of specialists (e.g., DBAs, support analysts) may find their career paths less clear. It may lead to technical debt if there is less use of SMEs. Skills may be hard to find and cross-training harder to achieve.

PORTFOLIO MANAGEMENT

Key Characteristic: Work is organized by empowering product leaders or teams to maximize the value of the product.

Benefits (Pros): This structure focuses on the value the products deliver, enabling a balance between projects delivering new capabilities and the operational work required to ensure services operate at proper levels of availability, performance, and security; that is, good trade-offs can be made to optimize the value of the product as a whole.

Drawbacks (Cons): Harmful competition may develop between product teams.

STRATEGY LINKAGE

Key Characteristic: IT becomes a clearer part of the primary value chain of the enterprise.

Benefits (Pros): IT is treated as a first class citizen for corporate strategy, and IT strategy is better aligned to business strategy.

Drawbacks (Cons): Although product leaders are autonomous, they must still align with corporate strategy; it may be difficult to create or implement a coherent IT strategy since much of the IT capability is distributed throughout the product teams.

SCENARIO: A MAJOR PRODUCTION INCIDENT REQUIRING A CODE FIX

Key Characteristic: The entire team is on-call, responsible for fixing production issues, especially when the incident was caused by a change.

Benefits (Pros): Teams delivering new features are accountable for quality. Fast feedback is possible to drive continuous improvement.

Drawbacks (Cons): Developers who would rather build new features than fix broken ones may be unhappy. Team burnout may occur if demand management is not carefully monitored and managed.

EVALUATION

🟩	= Promotes DevOps adoption
🟨	= Minor area of risk for DevOps adoption
🟥	= Major area of risk for DevOps adoption

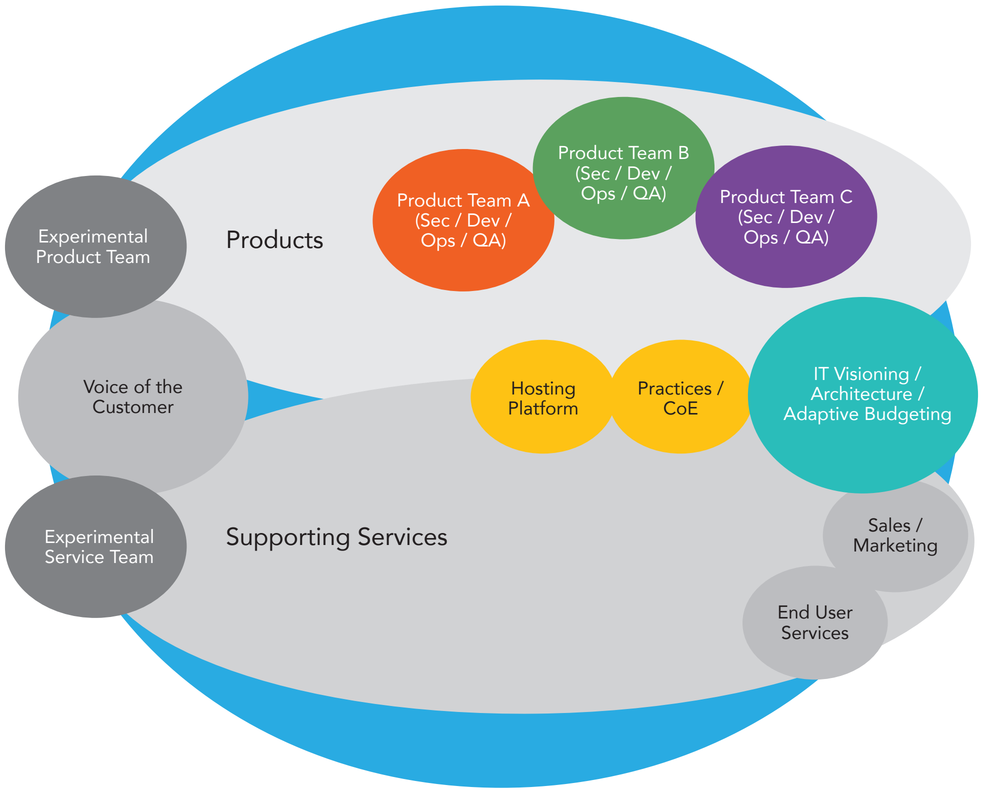
🟩	Promotes DevOps, short lead times
🟩	Encourages collaboration and communication (silos defeat communication)
🟨	Right kind of accountability (get on the right queue), routing instructions
🟩	Flat organization
🟩	Optimize the whole—deal with overall strategy, standards, etc.
🟥	Self-organizing overall structure—an it reconfigure itself easily?
🟨	Implements any necessary risk mitigation and compliance
🟥	Must fit into the rest of the organization

Ideas to Help Transition to a "Better" Model

This model assigns "capacity" to different products by associating teams with each product. It is somewhat "sticky" in that as needs change quickly, the capacity dedicated to each product might need to change, as might the skill sets and knowledge. Employees required for developing innovative products might be stuck in product teams and difficult to "repurpose." This problem can become especially severe when a product, once at the core of an organization's differentiating strategic offering, has evolved over time to become non-core or non-strategically differentiating. The mechanism that drives this situation is well-documented by Geoffrey Moore in his book *Dealing with Darwin*.[12] One solution to this problem is to establish an organizational model that is fundamentally adaptive, able to reconfigure itself in near real time to meet changing needs from a variety of stakeholders. We have named this type of organization "Model X" and have made an attempt to describe it below.

12. Geoffrey Moore, *Dealing with Darwin: How Great Companies Innovate at Every Phase of Their Evolution* (New York: Portfolio, 2008).

MODEL X—ADAPTIVE ORGANIZATION


Experimental Product Team
Products
Product Team A (Sec / Dev / Ops / QA)
Product Team B (Sec / Dev / Ops / QA)
Product Team C (Sec / Dev / Ops / QA)
Voice of the Customer
Hosting Platform
Practices / CoE
IT Visioning / Architecture / Adaptive Budgeting
Experimental Service Team
Supporting Services
Sales / Marketing
End User Services

While the three models described above (Functional Silo, Matrix, and Product and Platform) differ in many important respects, they share an important characteristic: they are all static or deterministic, with work governed through some kind of top-down control. What if, the structure of an organization was not static but fundamentally organic and adaptive? What if an organization wasn't governed by principles of "control?" Could an organization succeed without a single king-like figure at the top? With leadership and authority evenly distributed throughout? Wouldn't that allow decisions to be made quickly and effectively? What if budgeting occurred based on principles of abundance not scarcity? What if an enterprise could operate according to evolutionary principles and a more scientific understanding of what motivates people: autonomy, mastery, and purpose?[13]

In recent years, new models have emerged to enable organizations to be more nimble and adaptive to customer demands and real human needs. Just as Agile, Lean, and DevOps methodologies have removed friction and unproductive control structures from individual teams, it seems possible that more adaptive organizations could emerge at enterprise scale to better achieve speed, agility, security, and value.

In its purest form, such an organizational structure would be completely flat. To the extent that management remains (say, a CIO), it would focus much less on control and decision-making than more traditional models. Instead, management would focus on vision, culture, and people. Whereas in most organizations the CIO is treated as a chief, in the adaptive organization the CIO is more of a seer or healer, or, as Schwartz argues in *The Art of Business Value*, an interpreter of business value in this particular business context.[14] The product of management in such an organization is learning. Product teams take accountability and responsibility for setting their mission, staying aligned, and delivering "value" to customers and other stakeholders. This model contains the capability to generate (and retire) new structures to meet real-time needs from stakeholders as the organization learns. The structure of such an organization is fundamentally different from the prior three models. Rather than mimicking the structure of a feudal society (Model 1) or a machine (Model 3), Model X organizations look more like living organisms or ecosystems.

Salim Ismail in *Exponential Organizations* talks about organizations that are flexibly assembled "just in time" around projects or products based on customer demands and real human needs. He gives an example of a specific industry:

13. Daniel Pink, *Drive: The Surprising Truth About What Motivates Us* (New York: Riverhead Books, 2011).

14. Mark Schwartz, *The Art of Business Value* (Portland, OR: IT Revolution, 2016), 115-116.

> Today, Hollywood operates in exactly the same loosely coupled, networked environment of an Exo ecosystem. Each participant, from the writer and actor to the director and camera grip, manages his or her own career. Meanwhile, agents at every level help find and connect scripts with talent, production companies with equipment. These days, when a film is created, a swarm of independent entities come together for the duration of the production, operating on 24/7 schedules and in close collaboration. Once the film is finished, sets are broken down for re-use, equipment is reassigned, and all the actors, grips, and production assistants disband and scatter to pursue their next projects, which often start the very next day.[15]

While there are few models for building an organizational structure around these principles, such an organization would certainly be consistent with the DevOps approach.

KEY CHARACTERISTICS/BENEFITS/DRAWBACKS

ACCOUNTABILITY

Key Characteristic: In this type of organization, participants are aligned around a shared vision and purpose, and are jointly accountable for its success.

Benefits (Pros): The shared vision combined with flexibility promotes self-organization at an enterprise scale.

Drawbacks (Cons): This adaptive structure is new as a concept to enterprises. As a result, it may be threatening to many people, and accountability may seem unclear. It may work against stability.

15. Salim Ismail, *Exponential Organizations: Why New Organizations are Ten Times Better, Faster, and Cheaper Than Yours (And What to do About It)* (New York: Diversion Books, 2014), 116-117.

BUDGETING

Key Characteristic: A sense-and-respond approach to budgeting replaces yearly budget planning.

Benefits (Pros): An agile budgeting approach abandons attempts to predict and control, making the organization better able to identify and promote opportunities for "disruptive innovation."

Drawbacks (Cons): Short-term profitability may be sacrificed to achieve a long-term vision. It may present challenges in aligning with traditional accounting and financial requirements such as forecasting.

FLOW AND QUEUEING

Key Characteristic: The adaptive organization is entirely oriented toward flow.

Benefits (Pros): There are no organizationally imposed barriers to maximizing flow and minimizing wait time.

Drawbacks (Cons): Formal processes may not evolve, leading to unpredictability in time and quality.

PEOPLE

Key Characteristic: In the ideal situation, work is treated as a vocation, not just a job.

Benefits (Pros): The organization is built around people and focused on values and trust. Employees have a sense of working to achieve a higher purpose. This contrasts with siloed organizations that make empty claims that "people are our most valuable resource."

Drawbacks (Cons): It may be difficult to find the right SMEs for a particular problem. Organizations evolving to this model may find the culture shift too disruptive.

PORTFOLIO MANAGEMENT

Key Characteristic: Resources are allocated by hypothesis-driven development.

Benefits (Pros): Feedback loops in this model amplify learning to promote success and remove waste.

Drawbacks (Cons): Teams may have to spend lots of time in planning and prioritization meetings that was previously handled by a much smaller group of senior leaders and portfolio or program managers.

STRATEGY LINKAGE

Key Characteristic: This model allows the organization to decentralize strategic decisions and facilitates adaptive, responsive strategy-setting.

Benefits (Pros): By largely decentralizing strategy-setting, it is responsive to rapidly changing business needs and new insights. Strategy benefits from everyone's knowledge and insight.

Drawbacks (Cons): Without effective processes to align vision and mission across the enterprise, strategic goals can become incoherent and fail to support the enterprise vision. It may be more difficult for senior executives to communicate strategy to external stakeholders.

SCENARIO: A MAJOR PRODUCTION INCIDENT REQUIRING A CODE FIX.

Key Characteristic: The entire organization "swarms" to bring the people and resources together in real time to address the issue.

Benefits (Pros): Fast, effective reaction to address major incidents.

Drawbacks (Cons): If demand management is not carefully monitored and managed, other planned work may slow down as work is reprioritized to address the incident.

EVALUATION

🟩	= Promotes DevOps adoption
🟨	= Minor area of risk for DevOps adoption
🟥	= Major area of risk for DevOps adoption

🟩	Promotes DevOps, short lead times
🟩	Encourages collaboration and communication (silos defeat communication)
🟩	Right kind of accountability (get on the right queue), routing instructions
🟩	Flat organization
🟩	Optimize the whole—deal with overall strategy, standards, etc.
🟩	Self-organizing overall structure—Can it reconfigure itself easily?
🟨	Implements any necessary risk mitigation and compliance
🟩	Must fit into the rest of the organization

Of the models considered here, Model X is the most unusual in the context of modern organizations. Because there are few examples of this model in practice, it is less clear how organizations can transition into the model or what its challenges might be in practice. Conceptually, drawbacks to this model include the risk that without effective governance, teams may "sub-optimize the whole"—they may make decisions without considering the impact on other teams and the overall organizational system. Paradoxically, if teams adapt to their environment too quickly, speed and quality can suffer as teams always remain in the stages of forming, storming, and norming, and never achieve stability! Teams used to clear management authority structures may not easily make the transition to this model.

Challenges experienced by companies such as Zappos moving to Model X structures like Holacracy are well-documented, and the ability of this type of organization to succeed in the contemporary context is far from clear.[16] Whether these challenges are due to fundamental problems with the model itself or problems in making the transition from another organizational structure is similarly unclear. To some extent, evaluating this type of organization requires rethinking the very meaning of "success" given that many organizations making the transition to this model do not share the same values of profit and growth that are traditionally used as the barometer of success for most enterprises (at least those operating in the for profit marketplace).[17]

16. Jennifer Reingold, "How a Radical Shift Left Zappos Reeling," *Fortune*, March 4, 2016, http://fortune.com/zappos-tony-hsieh-holacracy/.

17. Ellen Huet and Brad Stone, "Silicon Valley's Audacious Plan to Create a New Stock Exchange," *Bloomberg*, June 12, 2016, http://www.bloomberg.com/news/articles/2016-06-12/silicon-valley-s-audacious-plan-to-create-a-new-stock-exchange. In this context, it will be very interesting to see how the recently announced effort by Eric Reis, Silicon Valley entrepreneur and author of *The Lean Startup*, to establish a new "long-term" stock exchange develops. Should this effort succeed, it is conceivable that enterprises could find more success adopting Model X with current pressures to focus on short-term profit above all else reduced if not eliminated.

ADDITIONAL CONCERNS WHEN CONSIDERING ORGANIZATIONAL DESIGN

While the structural model of the organization is important, there are other factors that play a role in determining how successfully the organizational structure supports DevOps. In this section we will raise a few issues that organizations need to consider as they transition to (or through) the models we have discussed in the preceding sections.

Bi-Modal IT

A recent study by Gartner proposed a Bi-Modal IT approach that divides IT into two groups that must necessarily move at different speeds:[18]

- Mode 1: Traditional IT that handles the methodical waterfall development and maintenance of critical legacy systems.
- Mode 2: Digital IT teams that employ Agile development and DevOps practices to quickly iterate on features and products.

Gartner's reasoning is that legacy systems are both too brittle and too

18. Simon Mingay and Mary Mesaglio, "How to Achieve Enterprise Agility With a Bimodal Capability," *Gartner.com*, April 24, 2015, http://www.gartner.com/it-glossary/bimodal/.

mission-critical to move into a higher velocity model of development. If Gartner is right, then we should consider building a separate organization for Mode 1 systems, perhaps one more in the traditional style, on the assumption that it will need to support a waterfall style.

However, there is increasing disagreement with this approach, and some believe that Bi-Modal IT is simply the latest incarnation of Taylorism, based on an underlying view of people as resources to be controlled and manipulated (see "Forget Two-Speed IT; DevOps Enables Faster Delivery Across The Board").[19] Left untreated, the legacy applications that are often core systems of record or capability, will fall behind as competition comes to provide the same capability with great adaptability and agility. As Jez Humble explains:

> "There are three serious problems with the Bimodal model which, when taken together, mean that leaders that fail to move beyond Gartner's advice will end up falling further and further behind the competition. They will continue to invest ever more money to maintain systems that will become increasingly complex and fragile over time, while failing to gain the expected return on investment from adopting agile methods."[20]

The problem with the Bi-Modal IT concept is that it denies legacy systems the advantages of fast-feedback, high-velocity development, to say nothing of the benefits of experimenting with different approaches to find what works best. As Markos Rendell puts it:

> "The creation of a faster Mode 2 is far less troubling to me *as a tactic*. If you want to make IT systems faster, starting with a few

19. Kurt Bittner, "Forget Two-Speed IT; DevOps Enables Faster Delivery Across The Board," *Gartner.com*, November 2015, https://www.forrester.com/report/Forget+TwoSpeed+IT+DevOps+Enables+Faster+Delivery+Across+The+Board/-/E-RES121391.

20. Jez Humble, "The Flaw at the Heart of Bimodal IT, Continuous Delivery," *ContinuousDelivery.com*, April 2016,https://continuousdelivery.com/2016/04/the-flaw-at-the-heart-of-bimodal-it/.

> systems and trying to make those go faster is a sound starting point and a great way to combat some of the learned helplessness that we all experience…Perhaps Mode 1 should be called "leave as-is for now" and Mode 2 should be referred to as "currently in-scope for a lot of experiments to uncover ways to improve."[21]

Our recommendation is to look at Bi-Modal IT's Mode 2 as a hypothesis that can be tested through experimentation, measurement, and learning, and that could ultimately be the pattern that would transform the entire organization to enable DevOps. Later in this paper, we will address different models that we see forming from this type of experimentation.

Budgets

THE PROBLEM

The budgeting process impacts both organizational design and the success of the DevOps transformation. It affects how decisions are made, how teams organized, how work is delegated, how projects are started and ended, and how responsibilities are allocated. The effect of the budgeting process, however, is indirect, and its importance is not often recognized when organizations think about transforming to DevOps.

The Traditional Functional Silo Model makes it easy to establish and enforce budget accountabilities. Dollars are allocated to the top of the hierarchy and passed down through the organizational structure; each manager is responsible for the spending of the people below him or her. Project investments are handled similarly, with funds spent at the discretion of the project manager, or distributed to others under the direction of the project manager.

---

21. Markos Rendell, "DevOps goes Bimodal: How to leverage Gartner's Mode 2 IT," *TechBeacon.com*, June 15, 2016, http://techbeacon.com/devops-goes-bimodal-how-leverage-gartners-mode-2-it. Italics added.

In the Traditional Functional Silo Model, the budget is one of the primary ways that managers “control” the activities of those who report to them. The top-down distribution of funds mirrors (or establishes) the top-down distribution of responsibilities. An important difference among the varieties of Model 2 organizations is the source of funding: Does it come from the functional groups or from the product groups? In Model 3, the funding comes through the product organization, which then has control over value decisions that weigh costs against benefits that can be delivered to customers.

But the very idea of an annual budgeting cycle, where the allocation of funds is fixed at the beginning of the year, is closely related to the waterfall approach: it assumes that the future can be known and planned for in a degree which is quite unlikely. It limits flexibility and agility, and substitutes the supposed knowledge of a group of experts for sensing, learning, experimentation, and adaptation throughout the year. It often results in the wrong conversations during the year: managers focus on whether the plan is being adhered to and on explaining any variations, rather than focusing on what has changed from expectations and how it can be accommodated by changing the spending plans. For an organization looking to be more nimble and responsive, budgets have exactly the opposite effect.

OTHER APPROACHES

To cope with more dynamic demands, some organizations are taking different approaches:

1. Shorter Cycles & Frequent Reviews
Some organizations are moving to a shorter budgeting and review cycle. The traditional construct of annually funding projects remain, but plans and allocations are revisited quarterly or on a rolling basis. Both business and IT continuously reprioritize what features to fund and collaborate on go/no go decisions.

The benefit of such a system is that it allows teams to dynamically allocate funds within their projects, encourages teams to think in terms of minimum viable products, and quickly corrects mistakes in allocations. There are, however, two major drawbacks. First, funds cannot be reallocated **across** projects without waiting for the next annual cycle. And second, if the quarterly review process is not streamlined, it can result in a constant process of budgeting in which managers end up spending more time jockeying for funds instead of doing real work.

2. Product Based Funding
Other organizations have been experimenting with product based funding, where applications and the associated resources are grouped into product teams and are given a share of the total budget. This budget is primarily managed by the chief product owner (who may be drawn from either the business or IT), who then further distributes it to his/her product owners. As with the previously described approach, shortening funding cycles, the chief product owner collaborates with the rest of the business to decide which activities to move forward with.The chief product owners have considerable flexibility to reallocate funds across both products and projects within their portfolio. And the product owners are responsible for maximizing value by delivering incrementally, starting with a minimum viable feature set, within their allocated funds.

The benefit of this approach is its simplicity. Instead of making bets on projects, CFOs just have to decide how the budget will be spread across various product groups, and then chief product owners and product owners have freedom to allocate their share as they see fit. As a result, budget decisions are pushed as close to the front line as possible, allowing for greater flexibility and improved market response times.

A prerequisite to this method is to move from the Traditional Functional Silo Model to a Project and Product model, which, as we know, is not easy. The drawback of the method is that product owners may have a tendency to fund the new/sexy work while ignoring routine maintenance. To overcome this issue, some organizations force teams to reserve a piece of the budget for maintenance and paying down technical debt.

3. Venture Based Funding
This is a method that a handful of companies are experimenting with. Like Silicon Valley-style venture capitalist firms, an executive board funds ideas for minimal viable products. Based on the effectiveness of the product and the learnings derived from it, additional funding is granted to the teams.

The goal of this method is to seed a number of ideas and fund only the ones that have experimentally been shown to work. The benefit is greater transparency and reduced financial risk. The drawback is that most companies are neither experienced nor culturally disposed to acting like venture capitalist firms. A "sink or swim" culture gives rise to other problems. It can create conflicting incentives: When viability is all that matters, why should employees follow any of the corporate rules, security policies, or infrastructure requirements? And if the idea is good, what prevents the developers from seeking actual seed funding and creating a new competitor?

4. Beyond Budgeting
A new concept in budgeting is called "Beyond Budgeting." With this approach, budgetary decision-making and empowerment is pushed directly to the front line. Managers and practitioners take responsibility for determining their own budgets. To help them make better decisions, employees are given a challenge (e.g., to deliver a product or feature within a certain time), given full insight into the company's budget and finances (including P&Ls), and then asked to determine a budget.

While the benefits of this model include empowerment, nimbleness, and the ability to have decisions made by those closest to market information, there are some very real challenges to implementing this model. Employees must become experts at understanding the financial statements in order to make smart decisions; security concerns must be dealt with; SEC reporting (if it is a publicly traded company) may become difficult or impossible;

and corporate espionage may be difficult to spot or take action against. To learn more about beyond budgeting, see the article from Beyond Budgeting Institute "The 12 Beyond Budgeting Principles explained by members of the Core Team."[22]

Changing the budgeting process is not an easy undertaking. There are significant risks and short-term disruptions to address as the new process takes hold. But because the budget affects both organizational structure and the flow of work into the DevOps teams, it must be taken into consideration in any DevOps transformation.

22. Beyond Budgeting Institute, "The 12 Beyond Budgeting Principles explained by members of the Core Team," *Bbrt.org*, accessed September 2016, http://bbrt.org/the-beyond-budgeting-principles/12-beyond-budgeting-principles-explained-members-core-team.

CONCLUSION: OPERATIONAL EXCELLENCE

Excellence is an art won by training and habituation. We do not act rightly because we have virtue or excellence, but we rather have those because we have acted rightly. We are what we repeatedly do. Excellence, then, is not an act but a habit.

—Aristotle

The goal of the IT organizational structure is to deliver operational excellence. But our understanding of what constitutes operational excellence has been changing over the last few years.

IT has historically been treated as a cost. Efforts to achieve operational excellence have focused on finding "efficiencies" to reduce cost. As IT services have become more deeply integrated with the enterprise's core product and service offerings, they have become central to the enterprise's strategy for innovation, and must be adapted to meet changing market conditions and opportunities with ever increasing speed and agility. This new reality demands that we adopt new perspectives on both *operations* and *excellence* for IT organizations and enterprises overall. As Jim Highsmith, one of the authors of the Agile Manifesto, has recently argued, a new definition of operational excellence is required that focuses less on excellence in *operational efficiency* and more on excellence in *operational agility*.[23]

23. Jim Highsmith, "Redefining Operational Excellence in the Digital Age," *Thoughtworks.com*, June 1, 2015, https://www.thoughtworks.com/insights/blog/redefining-operational-excellence-digital-age.

Regardless of its organizational design, an IT organization should consider being more *efficient* (that is, doing things "right") as an *outcome* of its work to achieve excellence. Achieving excellence should instead be more concerned with helping the organization learn to be more *effective* (doing the right things). Adopting this new perspective on operational excellence aligns the DevOps movement to the demands of corporate strategy and governance.

Achieving operational excellence in the context of DevOps goes beyond juggling the categories of people, processes, and technology typically used to describe organizational systems—it also requires focus on culture and leadership. The culture of many organizations prizes execution according to plan over nimbleness, quality, and customer experience. Without cultural transformation, they will find themselves continuing to move ever more slowly under the weight of all the customer experience issues, technical debt, and critical incidents that take time away from new value delivery.[24]

The transformation to a DevOps organizational structure is necessarily a transformation to a culture where leaders provide vision and employees are empowered.

When we speak of people as resources, we imply they are expendable and interchangeable; in truth, they are neither. Not process, not technology, but *people* hold the keys to success for any DevOps transformation. The primary function of leadership must be to develop the capabilities of our employees and the environment in which they operate to allow them to think and act more independently, experiment, be innovative, learn, grow, improve, and achieve a greater sense of purpose. For an organization to develop everyone at all levels as leaders…that would be the ultimate sign that operational excellence had been truly achieved.

24. It is very interesting to consider that the 2016 State of DevOps research team discovered that high performing organizations are 2.2x more likely to recommend their organization as a great place to work. In other words, changing our perspective on excellence to focus less on efficiency and more on agility will have a significant side-benefit of improving employee engagement at the same time that we are better able to achieve our corporate objectives, especially those around availability and customer experience.

IT Strategy Setting

DevOps makes it clear that IT must be considered as a whole: it makes apparent the entire value chain that is involved in IT value delivery. The coherence of the entire IT value chain should be addressed as more than just an operational issue. DevOps is more than just creating cross-functional teams to execute projects—it requires that the corresponding organizational structures, or functional areas, be integrated to support a unified IT strategy. We can no longer optimize and innovate within functional silos—investing separately in *application innovation* and *infrastructure innovation*. In a DevOps world, these are one and the same.

As a consequence, the IT organizational structure must not only bring together the different functions in execution, or tactically, but must support strategy setting across the IT value chain. The models described in this paper, again, are only a piece of the structural transformation that is necessary to support DevOps. A successful organization will also consider the effect of the organizational structure on strategy setting, budgeting, and operational excellence.

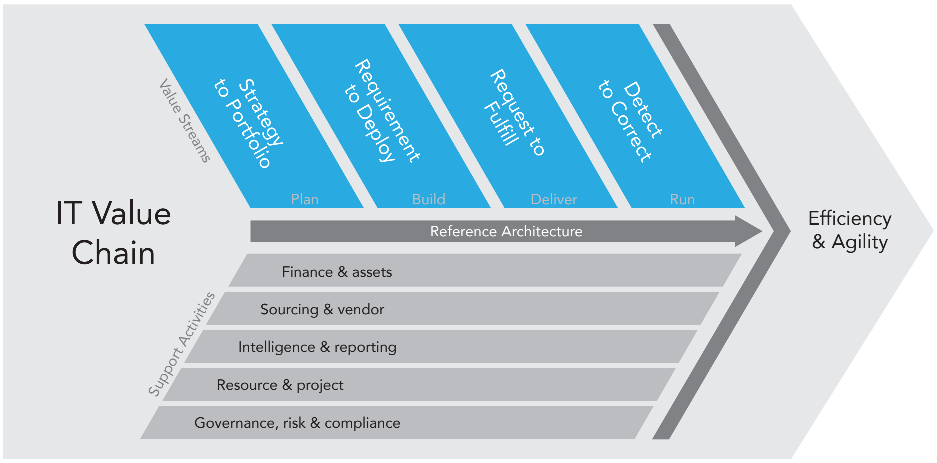


Figure 5: Source: "IT4IT™ Overview" The Open Group website, accessed September 2016, http://www.opengroup.org/IT4IT/overview.

Learning How to Learn

The structure of an organization can profoundly help or hinder the use of DevOps practices. That said, any effort to promote DevOps based on organizational design alone is bound to show limited results, if not fail outright. More than anything, the principles, practices, behaviors, and mindsets characteristic of a DevOps organization lie in the culture of the organization, not in its documented formal structure. What, then, is the best way to drive cultural change and ensure a new organizational design is successful?

While we cannot offer a definitive answer to this question—since "best practices" have not yet crystallized—we can provide a *clear warning*: where change requires tinkering with organizational culture, formal programs alone are not effective. These tend to be driven by strategic and budgetary planning cycles with clearly delineated beginnings and endings. Sadly, what frequently results is little more than the superficial appearance of change and premature declarations of completion and success.[25] Formal initiatives may be helpful in solving complicated problems where the scope of the problem and solution are well understood (e.g., a program to migrate a set of applications from a datacenter to a cloud-based environment). Cultural change, on the other hand, is not just *complicated*, but—in the terminology of Systems Theory—inherently *complex*. In other words, the behaviors that result from making a "change" to culture are "emergent properties" of the organizational system—relationships between cause and effect cannot be fully predicted or planned, and only somewhat controlled.[26]

25. Alfred Korzybski, "Map–territory relation," *Wikipedia.com*, last modified July 29, 2016, https://en.wikipedia.org/wiki/Map–territory_relation. Much as with local government entities, struggling to effectively address air and water pollution in ways that fail to recognize the inherent limitations of man-made physical boundaries of city, county, state, and country, organizations may find that culture and its transformation has very little respect for programs and initiatives artificially-bound by corporate fiscal calendars and annually re-imagined strategic planning cycles. In this way, Alfred Korzybski's dictum that "The map is not the territory" applies as much to geographies of time as space.

26. For more information on Systems Theory, see the work of David Snowden.

In the context of DevOps, organizational change management presents some unique challenges, in that its focus is specifically on the two cultures of Dev and Ops. When it comes to large-scale organizational restructuring, however, there is a third culture that must be attended to: the culture of executives. This is well described by Edgar Schein in his 1996 article, "Three Cultures of Management: The Key to Organizational Learning."[27] Schein asks, "Why do organizations fail to *learn how to learn* and therefore remain competitively marginal?" Considering the emphasis in DevOps on feedback loops to drive continuous improvement, Schein's article seems especially relevant. How can learning be built into the very structure of the organization itself, and be used to determine the best structure?

Schein emphasizes three points:

1. The significance of organizational culture(s) is underappreciated.
2. Assumptions about how to drive change are no longer valid in the twenty-first century.
3. New forms of communication ("channels") are required to help the members of each organizational culture learn from one another.

Any re-structuring of an organization is, at its core, a re-structuring of people and their relationships of authority and accountability. Since DevOps is about improving collaboration between people and helping them learn how to solve common problems for the benefit of their shared stakeholders, we can only recommend a collaborative approach to change. To the extent that there is anything resembling a "best practice" for nurturing teams and helping desired changes in organizational structure take hold and thrive, it is that adopting a collaborative approach is most likely to be effective.

27. Edgar H. Schein, "Three Cultures of Management: The Key to Organizational Learning," *Sloan Management Review*, 38 no. 1 (Fall 1996).

To summarize our findings:

- There is no single "best design" for an organization that wishes to adopt DevOps.

- Especially at an enterprise scale, if care is not taken, changing an organization's structure for DevOps may simply result in new silos, just along different dimensions.

- We need to question the assumption that a design should always be determined up-front, implemented as designed, and rigidly maintained over the long-term.

A "good" DevOps organization is one in which the organizational structure can be adapted over time to address problems as they are better understood, as well as reinforce and scale solutions that may emerge in pockets of the organization. Our change management approach, then, must go beyond driving acceptance of a predefined organizational structure—it must shift people's mindsets to accept continuous or repeated structural change over the long term. Such an approach is supported by a recent article published by the Lean Enterprise Institute, in which change management is connected to the lean principles of experimentation and continuous improvement through the application of PDSA and the nurturing of a growth mindset across the organization.[28]

It may be useful for organizations to borrow some concepts and practices from Model X, even if if the organization is structured in a "less mature" model such as Model 2. Promoting a growth mindset, connecting continuous improvement to a long-term commitment to evolving the structure of the

28. Katrina Appell, "The Problems Inherent in Change—And What You Can Do About Them," *Lean.org*, March 29, 2016, http://www.lean.org/LeanPost/Posting.cfm?LeanPostId=554.

organization, may be the best approach to achieving the "perfect" structure for any organization.[29]

The organizational structures described in this article are just models—while some actual organizations do resemble these models very closely, others are hybrids or variations. Finding the right model for your organization is a complex process, and we hope this paper is helpful to you on your DevOps journey. We wish you the greatest success in your endeavours, especially where those endeavors align to the Lean principles of DevOps and a focus on the never ending work to improve the work of your organization.

Finally, we return to where we started, with the quote from retired US Navy Captain David Marquet: "Fix the environment, not the people." To make DevOps work as a global society, we must learn over time how to treat our organizations as complex, living ecosystems functioning in an even more complex living environment: planet Earth. We will not improve anything by trying to "fix the people," for they are not broken. What's more, we would like to suggest that our environments similarly do not really need to be fixed: they need to be healed.

29. In this context, it is worthwhile noting that the Lean principle of seeking perfection does not imply that perfection is ever achieved or even completely understood. In this way, an organization's very notion of perfection itself needs to be subject to a process of continuous improvement.

References

Appell, Katrina. “The Problems Inherent in Change—And What You Can Do About Them,” *Lean.org*. March 29, 2016. http://www.lean.org/LeanPost/Posting.cfm?LeanPostId=554.

Arandjelovic, Pedja, Libby Bulin, and Naufal Khan. “Why CIOs should be Business-Strategy Partners.” *McKinsey.com*. February 2015. http://www.mckinsey.com/business-functions/business-technology/our-insights/why-cios-should-be-business-strategy-partners.

Beal, Helen. “8 Correlations Between DevOps and Holacracy.” *Ranger4*. July 8, 2016 http://www.ranger4.com/blog/8-correlations-between-devops-and-holacracy.

Beyer, Betsey, Chris Jones, Jennifer Petoff, and Niall Richard Murphy, eds. *Site Reliability Engineering: How Google Runs Production Systems*. (Sebastopol, CA: O’Reilly Media, 2016).

Beyond Budgeting Institute. “The 12 Beyond Budgeting Principles explained by members of the Core Team,” *Bbrt.org*. Accessed September 2016. http://bbrt.org/the-beyond-budgeting-principles/12-beyond-budgeting-principles-explained-members-core-team.

Bittner, Kurt. “Forget Two-Speed IT; DevOps Enables Faster Delivery Across The Board.” *Gartner.com*. November 2015. https://www.forrester.com/report/Forget+TwoSpeed+IT+DevOps+Enables+Faster+Delivery+Across+The+Board/-/E-RES121391.f.

Conway, Mel. “Conway’s Law.” *MelConway.com*. Accessed September 2016. http://www.melconway.com/Home/Conways_Law.html.

Deming, W. Edwards. 1994. *The New Economics*. (Cambridge, MA: MIT Press, 1994).

Deming, W. Edwards. “The PDSA Cycle.” *The W. Edwards Deming Institute website*. November 28, 2012. https://www.deming.org/theman/theories/pdsacycle.

Drnevitch, Paul and David Croson. “Information Technology and Business-Level Strategy: Toward an Integrated Theoretical Perspective.” *MIS Quarterly* 37, no. 2 (June 2013).

Highsmith, Jim. “Redefining Operational Excellence in the Digital Age.” *Thoughtworks.com*. June 1, 2015. https://www.thoughtworks.com/insights/blog/redefining-operational-excellence-digital-age.

Huet, Ellen and Brad Stone. “Silicon Valley’s Audacious Plan to Create a New Stock Exchange.” *Bloomberg.com*. June 12, 2016. http://www.bloomberg.com/news/articles/2016-06-12/silicon-valley-s-audacious-plan-to-create-a-new-stock-exchange.

Humble, Jez. “The Flaw at the Heart of Bimodal IT, Continuous Delivery.” *ContinuousDelivery.com*. April 2016. https://continuousdelivery.com/2016/04/the-flaw-at-the-heart-of-bimodal-it/.

Ismail, Salim. *Exponential Organizations: Why new organizations are ten times better, faster, and cheaper than yours (and what to do about it)*. (New York: Diversion Publishing, 2014).

Kelly, Alan. "No Projects—Beyond Projects." *InfoQ*, December 5, 2014. https://www.infoq.com/articles/kelly-beyond-projects.

Kim, Gene, Patrick Debois, John Willis, and Jez Humble. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. (Portland, OR: IT Revolution Press, 2016).

Laloux, Frederic. *Reinventing Organizations*. (Brussels: Nelson Parker, 2014).

MacCormack, Alan, John Rusnak, and Carliss Baldwin. "Exploring the Duality Between Product and Organizational Architectures: A Test of the "Mirroring" Hypothesis." *Research Policy* 41, no. 8 (October 2012).

Mingay, Simon and Mary Mesaglio. "How to Achieve Enterprise Agility With a Bimodal Capability." *Gartner.com*. April 24, 2015. http://www.gartner.com/it-glossary/bimodal/.

Moore, Geoffrey. *Dealing with Darwin: How Great Companies Innovate at Every Phase of Their Evolution*. (New York: Portfolio, 2008).

Pink, Daniel. Drive: *The Surprising Truth About What Motivates Us*. (New York: Riverhead Books, 2011).

Project Management Institute. "Capturing the Value of Project Management." *Pmi.com*. February 2015. http://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2015.pdf

Reingold, Jennifer. "How a Radical Shift Left Zappos Reeling." *Fortune*. March 4, 2016. http://fortune.com/zappos-tony-hsieh-holacracy/.

Rendell, Markos. "DevOps goes Bimodal: How to leverage Gartner's Mode 2 IT." *TechBeacon.com*. June 15, 2016. http://techbeacon.com/devops-goes-bimodal-how-leverage-gartners-mode-2-it.

Schein, Edgar H. *Organizational Culture and Leadership*. (San Francisco: Jossey-Bass, 1985).

Schein, Edgar H. "Three Cultures of Management: The Key to Organizational Learning." *Sloan Management Review*, 38, no. 1 (Fall 1996).

Scholtes, Peter R. *The Leader's Handbook*. (New York: McGraw-Hill, 1997).

Schwartz, Mark. *The Art of Business Value*. (Portland, OR: IT Revolution Press, 2016).

Skelton, Matthew. "Introduction." *DevOpsTopologies.com*. accessed September 2016. http://web.devopstopologies.com.

Velarde, Felix. "Is Holacracy finally dead?" *Quartz*, May 16, 2016. http://qz.com/677130/is-holacracy-finally-dead/.

Wikipedia. "Conway's law." Last modified September 14, 2016. https://en.wikipedia.org/wiki/Conway%27s_law.

Authors

Mark Schwartz, CIO, US Citizenship and Immigration Services, mark.schwartz@gmail.com

Jason Cox, Director Systems Engineering, Disney, jason.cox@disney.com, @jasonacox

Jonathan Snyder, Sr. Manager, Service Deployment & Quality, Adobe Systems, jsnyder@adobe.com

Mark Rendell, Principal Director, Accenture, mark.rendell@accenture.com, @markosrendell

Chivas Nambiar, Director Systems Engineering, Verizon, chivas.nambiar@one.verizon.com

Mustafa Kapadia, NA DevOps Service Line Leader, IBM, m.a.kapadia@us.ibm.com

OTHER CONTRIBUTORS

Alyson Hoffman, IT Revolution, alyh@itrevolution.net