The Checkbox Project

Learnings for Organizing for Outcomes

Kamran Kazempour,
Chris Hill, Steve Pereira,
Dean Leffingwell, Amy Willard,
and Gene Kim

FOREWORD FROM THE PUBLISHER

For the first time since the COVID-19 pandemic hit, we were able to hold the annual DevOps Enterprise Forum in person again. For nine years, this forum has brought together the brightest minds in technology leadership and new ways of working that meet the unique challenges of the digital age. Once again this year, we were delighted that we had two groups from the US Department of Defense to discuss the unique challenges the armed forces face in meeting the technological challenges of today and tomorrow.

Over the course of the three days, this group worked to identify and create written guidance for the top problems facing the technology leadership community, including how to talk business as a technology leader, measuring value, software asset inventory, and so much more.

My thanks go to Jeff Gallimore and Ann Perry for their continued support and assistance in organizing this event and making it such a success.

And finally, a huge thank-you to the 2023 DevOps Enterprise Forum participants, who contributed their time, expertise, creative energy, and professional experiences with the group and with the wider DevOps community. Our participants always go above and beyond to put together these resource guides for the entire community to share, learn, and improve the state of technology and the industry.

—Gene Kim
September 2023
Portland, Oregon

Author
Kamran Kazempour
Software Delivery & Developer Experience, Wallet, Payments and Commerce, Apple

Chris Hill
Senior Member of Technical Staff, Director, T-Mobile

Steve Pereira
Lead Consultant, Visible Consulting

Dean Leffingwell
Cofounder and Chief Methodologist at Scaled Agile, Inc.

Amy Willard
IT Director, HR, LR, Corp Functions, Strategy & Transformation, John Deere

Gene Kim
Researcher and Coauthor of *Wiring the Winning Organization*

Introduction from Gene Kim

Let us start with the observation that winning organizations are able to do extraordinary things, more than any single individual could ever do alone. A winning organization fully unleashes people's creativity and capabilities. These organizations generate more value, in less time, at lower cost, and seemingly with less effort.

Contrast that to an organization at the opposite end of the performance spectrum, which somehow constrains or even extinguishes the creativity and problem-solving capabilities of people. They generate less value, over more time, at higher cost. And every activity, no matter how small, requires an incredible amount of heroic effort from a vast number of people.

What is remarkable is that the two organizations at the opposite ends of the performance spectrum can be the same organization. They are identical in every way except for how they are wired.

This paper is one of the more rewarding efforts I've been a part of. It presents a startling case study of how a seemingly small effort, one which should result in a huge amount of business value, is nearly impossible to complete. It requires an epic amount of communication, coordination, escalation, prioritization, synchronization, and deconfliction across almost every functional silo in the organization. And despite the heroic efforts across multiple quarters of effort, the results are still not the best quality and do not deliver the value promised.

This story will probably be startlingly familiar to anyone working in a large, complex organization. And although the proposed solution might be insightful and obvious, I suspect it will require a significant amount of rethinking of how we truly wire our organizations to win, at a scale far broader than most of us have ever undertaken before.

I applaud this group's effort, and I think it is one of the most important papers to come out of the DevOps Enterprise Forum.

Gene Kim
Portland, OR
June 2023

The Case Study

How long does it take to enable one checkbox? A seemingly straightforward digital task; yet, for a large enterprise, it often proves to be anything but.

Consider a company with a user base in the millions, which we will call Parts Unlimited. Their business team wanted to add an additional monthly third-party subscription service. Their users would simply have to check "Yes" to opt-in. The service offering was a third-party pass through, requiring no additional delivery from Parts Unlimited.

However, adding this recurring subscription reveals a complex web of interdependencies. Product development needs to understand customer journeys. The IT department has to manage technical aspects. Global legal and compliance teams must ensure privacy regulations are met. Finance needs new fields for tracking revenue and for tax purposes. Marketing must determine segmentation and drive campaign promotion and success. And on and on and on across multiple channels and lines of business (Figure 1).

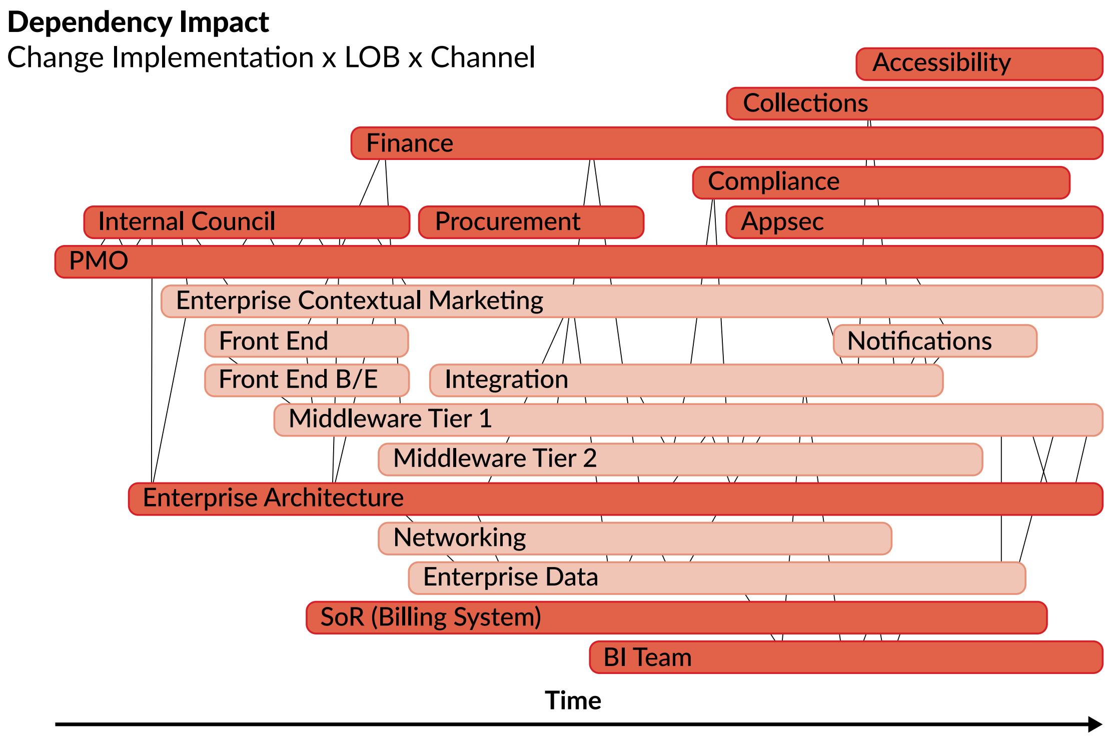


Figure 1: Known and Discovered Dependencies for Implementing Checkbox
Each dependency was tightly coupled and required exhaustive coordination.

This single opt-in checkbox stumbled over nearly all of the challenges that come when cross-organizational collaboration is required to deliver value. Parts Unlimited wanted to go fast, but it was just not organizationally equipped to expedite and facilitate a working implementation in a rapid time frame. Doing so required that they criss-cross organizational silos to deliver a checkbox that provided recurring revenue and economic value for the company.

For Parts Unlimited, there were three primary stages in implementation:

1. **Experience iterations:** Understanding the user journey and discovering the permutations across various teams, value streams, and business segments.
2. **Design iterations:** The technical implementation required determining where the code would live across the system of records and technology stacks, and how data would be structured and flow.
3. **Program activities:** Planning, funding, and delivering this initiative across multiple boundaries did not flow naturally. Substantial program management personnel and activities were required.

Let's take a deeper look at these stages in practice.

First, Experience Iterations

The subscription checkbox soon starts to materialize as more complex than originally thought. User experience and business model questions demanded immediate attention. Should this offering go to all of their customers? Are there any customers for which it's not applicable? Should customers have permission to approve the service on behalf of all users in their account? What monitors and resolutions will be in place if the integration is struggling or fails?

No one person or team can make these decisions on their own. Experience iterations span vertically down the IT teams and architects, and horizontally across multiple major P&L's across the entire company. Each experience iteration requires new code and artifacts to be created by individuals and realigning and context-switching from previously existing priorities, previous commitments, attention to security and compliance—all before any customer or company value can be created.

Then, Design Iterations

Next came design iterations, and the spider web of dependencies continued to expand. In terms of customer interfaces, how will the checkbox be displayed across multiple front-ends? In regards to billing, can their existing system handle the number of additional calls to check and write to the checkbox store? How will this be monitored and operated? Are there layers in place to ensure duplicative actions don't occur? What about payment and non payment issues?

All design iteration scenarios would have to be thoroughly tested, but there were many challenges. Which team would generate accurate testing and customer data? How can changes be made to the system with no downtime or loss of transactions? Increasingly, they were also challenged with the challenge of enticing users to opt-in. Questions, decisions and dependencies continued to mount, and they involved an expanding number of teams.

Finally, Program Activities:

Due to the increasing scope of this project and the time it would require, a centralized program management function stepped in to ensure the workflow was organized. This included defining which teams need to do what work, what their current priorities and new schedule estimates would be, and how much it would cost the teams to make this "simple checkbox" change. The program managers at Parts Unlimited led priority conversations across more than twenty separate engineering teams and uncovered a substantial list of dependencies.

No tooling or representation for dependency impact existed, so Figure 1 is just a rough approximation of the tangled web of dependencies, many of which only became apparent when the project reached an impasse. In addition to the these, they also had to:

- Determine go-no-go dates and iterations with "big bang" releases.
- Account for architectural changes as they learned more.
- Manage costs and OPEX conversations with every involved team for supporting this functionality.
- Create contracts and agreements with vendors, publish RFPs, hold vendors accountable, engage legal on all contracts and agreements, and accommodate for scope changes and increases in budgets.

The Results

In the end, Parts Unlimited achieved what they wanted, the ability to activate a "simple checkbox" so users could opt-in to an additional third-party subscription and enhance both customer and enterprise value. Doing so, however, required a managed and close collaboration across over twenty teams in multiple organizational hierarchies across multiple channels and segments, including involvement from many coordination roles and shared services (Figure 2). In addition, it required involvement of senior level executives to handle reprioritization of current tasks, rally staff, and handle escalation.

In the end, the project took over twelve months from conception to completion, and cost the company over $28 million to implement. Few stakeholders would consider it a success.

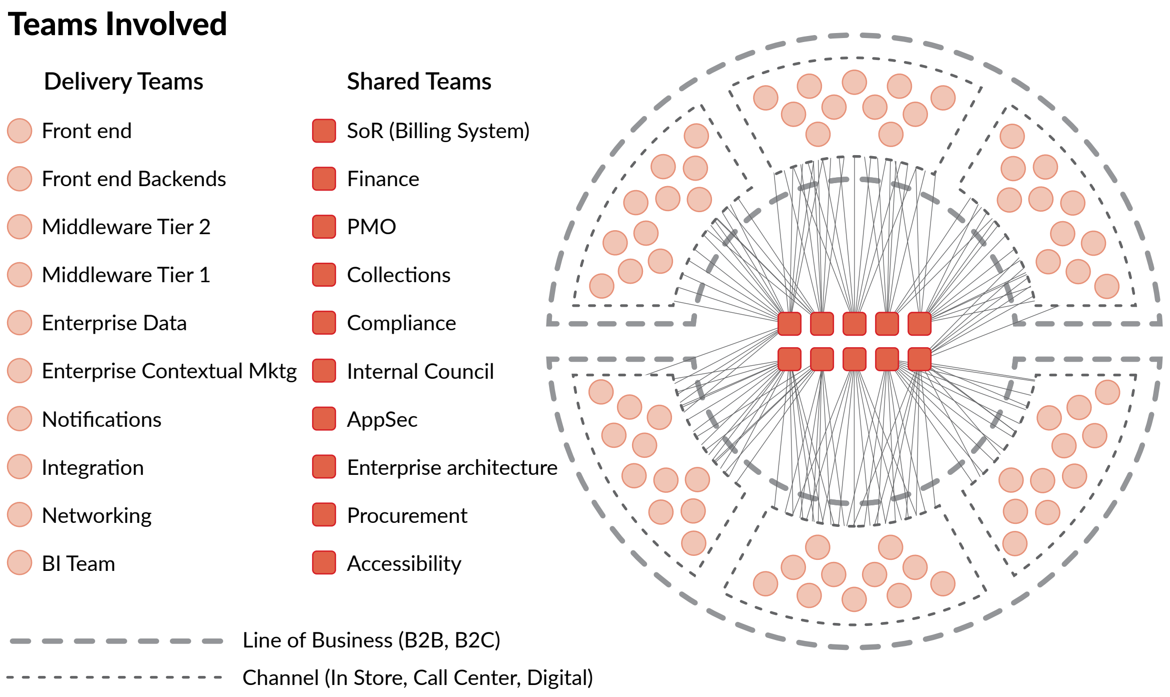


Figure 2: Teams Involved in Checkbox Project

The effort spanned 10+ delivery teams for each of 2 lines of business (LOB) and 3 channels for each LOB, each heavily reliant on shared services.

Evolving Your Organization toward Better Outcomes

The Checkbox Project reveals the costs of an organizational gap (i.e., an organizational structure that prevented fast flow of new value). One contributing cause is the manner in which large enterprises typically use reporting hierarchies as an anchor point for individual agencies. After all, if you don't know who you are reporting to, how do you know where you belong or what you are supposed to do?

There is often a PowerPoint diagram that shows each name in a tree-like structure. It provides a collective perspective of who fits where. Ideally, this depicts the most-efficient structure for future work. However, what is efficient for reporting and accountability isn't necessarily efficient (or effective!) for customer value delivery. The functional orientation of typical organizational structures creates silos that are challenging to see or interact beyond (as in Figure 3), despite many functions being required to deliver a desired outcome.

But future work could touch every team in your entire company or a single team or somewhere in the middle. Ideally, the existing structures yield the shortest lead time for a new scope. Unfortunately, there are many dimensions that could impede the ideal state. The flow of work in many large organizations follows a meandering path through the organization (as in Figure 4) and often even doubles back on itself (or wanders in circles) before making progress toward delivered value.

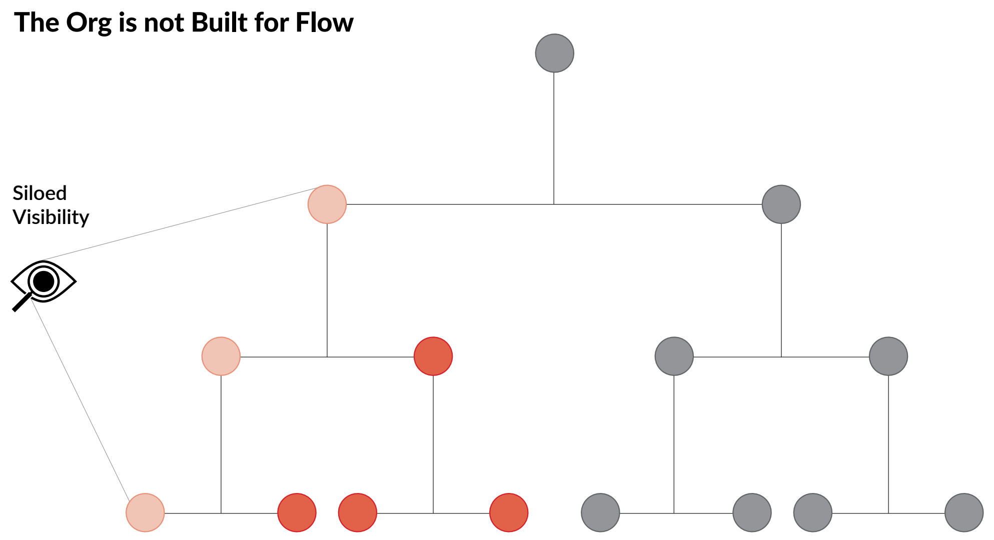


Figure 3: Typical Org Chart

The org chart is typically functional, not aligned to customer needs.

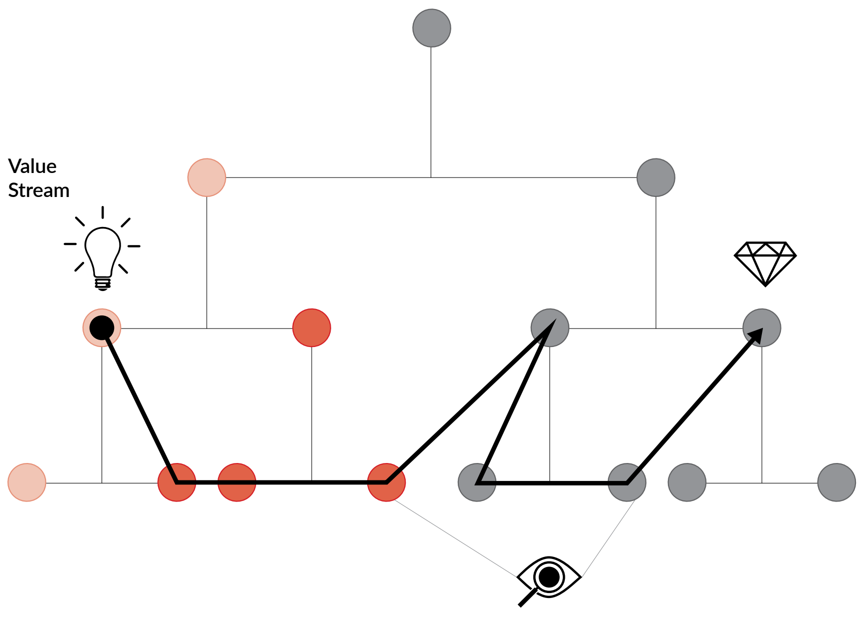


Figure 4: Actual Path in the Org Chart

The path to customer value often stretches beyond any individual view of the organization.

The following section will provide some ideas on how the Checkbox Project could be improved upon in future iterations. Let's face it, there's always another project! While there is certainly no "one right way" to organize for better outcomes, the tips below may help you on your way.

Foster Shared Visibility and Build Clarity

Part of the challenge of the Checkbox Project was the lack of visibility into what it takes to deliver what seemed to be a simple change. Beyond that, a major downstream issue arose. Every team involved in the change was alerted to their involvement in the midst of their existing backlog. Not only were there delivery costs, but there were also opportunity costs to consider. And, there are social costs! What's the cost of delaying your current road map to accommodate a change from another division? What's the cost of delaying twenty individual road maps and the commitments they entail?

To address this, consider the following:

- **Map out the current landscape.** Visualizing is a powerful thinking and collaboration tool, even if it is just a collection of post-it notes[*] with all the teams involved in a particular effort. When you start to look at the big picture, you can begin to connect the dots that drive powerful conversations, insights, and decisions.
- **Create a shared understanding of the goal.** The successful achievement of any initiative hinges upon creating a shared understanding of the goals among all stakeholders involved and affected. By gathering stakeholders to share ideas, concerns, and perspectives, you can uncover surprising collective insight, clarity, and support.[†] Tools like video recording, online white-boarding, or textual documentation create a future reference point that can be easily shared up, across, and down the organization.
- **Map the value stream.** Mapping your value stream[‡] is one strategic way of providing a shared view of the current state, constraints, and potential future flow of your operations.[§] The data derived from value stream mapping can lead

* Steve Pereira and Andrew Davis, "Flow Engineering," IT Revolution (blog), June 21, 2022. https://itrevolution.com/articles/flow-engineering.

† Steve Pereira, "Outcome Mapping - How to Collaborate with Clarity," InfoQ, July 23, 2021. https://www.infoq.com/articles/outcome-mapping-clarity/.

‡ " Value Stream Mapping: What It Is + How It Relates to DevOps," Pluralsight (blog), March 10, 2023. https://www.pluralsight.com/blog/it-ops/value-stream-mapping.

§ Gene Kim, Jez Humble, John Willis, and Patrick Debois, "Understanding the Work in our Value Stream and Improving Flow," IT Revolution (blog), November 30, 2021). https://itrevolution.com/articles/improve-flow-devops-value-stream/.

to profound realizations, can identify points of unnecessary or potential friction and waste, and can reveal clear opportunities for improvement. Additionally, the process of mapping and visualizing both organizational and technical dependencies encourages productive discussions and informed decisions.

- **Manage the entire value stream.** Investigate Value Stream Management[*] as a large-scale flow improvement enabler. Lean practices and principles work well for large organizations looking to simplify and deliver value with less disruption. Large scale efforts can be holistically measured, predicted, and even simulated to improve operational efficiency and effectiveness while minimizing risk.

Optimize Team Structure by Balancing Size and Autonomy

The Checkbox Project demanded extensive communication, coordination, and collaboration. Having every required contributor on one team is never an option, but neither is a multitude of silos. With millions on the line and dozens of teams at play, it pays to invest in organizational structure that facilitates agility and quicker time to market.

- **Leverage resources and interaction modes**[†] that replace face-to-face coordination, consensus, and consultation. Each of the blocking decisions that stopped the Checkbox Project from proceeding could have been assisted by decision trees or filters, frameworks, documentation, and principles that guide decision-making and tactics. Of course, you can't think of everything in advance and there are diminishing returns (and maintenance costs!) for resources, but there is great value in minimal investments and doubling down where opportunities prove fruitful.
- **Applying Team Topologies**[‡] can help clarify cross-cutting accelerators, like an enabling team for partner offers or the creation of a platform team or complicated subsystem team for facilitating billing changes.
- **Achieving sustained performance and value delivery** requires a decisive product owner who has the authority to make strategic decisions. Amazon's Single-Threaded Model[§] establishes and supports decision rights and owner-

[*] Saahil Panikar, Cindy Van Epps, Jeffrey Shupack, "Value Stream Management and Organizing around Value," IT Revolution (blog), June 14, 2022. https://itrevolution.com/articles/value-stream-management-and-organizing-around-value.

[†] Andrew Harmel-Law, "Scaling the Practice of Architecture, Conversationally," MartinFowler.com (blog), December 15, 2021. https://martinfowler.com/articles/scaling-architecture-conversationally.html.

[‡] "The Four Team Types from Team Topologies," IT Revolution (blog), February 28, 2023. https://itrevolution.com/articles/four-team-types/.

[§] Pedro Del Gallego, "Single-Threaded Leaders at Amazon," PedroDelGallego.github.io (bog), January 2, 2022. https://pedrodelgallego.github.io/blog/amazon/single-threaded-model/.

ship (along with accountability) at the team level. This aligns with the principle of pushing decision-making capabilities down to the level where they can be most effectively deployed, but also where the context is most relevant and available. The result is low-latency, high-signal decision-making and action.

- **One approach that aids this autonomy** (and loosely-coupled composability with other teams) is to think of your teams as APIs. APIs are accessible, easily leveraged, independent, and composable. By emulating these properties within your team structure, their capabilities become clear and easily accessible to dependent teams without necessitating costly direct, synchronous interactions. This approach, as exemplified by Amazon within AWS, has proven highly successful.
- **Reducing the need for coordination between teams** also helps to ease cognitive load and distractions. Implementing service catalogs can help define and clarify roles, responsibilities, and operations across teams. Team APIs* create clarity and visibility around team interactions, capabilities, and dependencies. A service catalog can start to create definition around teams and interactions with minimal up-front investment.

Architect for Agility

System architecture is either as decentralized or decoupled as the organizations that drive it. The disruptive "all hands on deck" and deep coordination called for by the Checkbox Project can be avoided when each team can autonomously accommodate a change according to their schedule and priorities without impeding dependent teams.

You may need a database change to fully implement a change, but if all that's required of your team is an API call, you can deliver and move on. Additional opportunities to architect for agility include the following:

- **Serverless options** reduce requirements for ground-up infrastructure investments, while allowing for fast experimentation and design changes.
- **Event-driven architecture** enables loosely coupled systems through publishing, subscribing, and reacting to events or "state changes." This facilitates quick changes and new features with minimal risk and little need to coordinate across publishers and subscribers.
- **API-first architecture** simplifies interfaces and interactions, allowing teams to develop more and faster without deep coordination or dependencies.

* Get a team API template here: github.com/TeamTopologies/Team-API-template.

- **Domain-driven design** establishes clear domain boundaries in a product-oriented and agile model. It also fosters a common language for building comprehension among both technical and non-technical team members. More clarity and definition means more action without getting everyone in a room to hash things out when."
- Platform engineering* practices and product thinking for platforms are rapidly evolving. There are minimal starting points,† good practices,‡ and capable platform-as-a-service offerings. Shared platforms (illustrated in Figure 5) reduce tight coupling and the interrupt-driven model of shared teams, where tickets and queues dominate over self-service access to capabilities. Parts Unlimited could have avoided significant dependency delays by investing in the ability for delivery teams to access platform capabilities to perform actions like adding data fields, events, producers, and consumers of changes like checking a box.

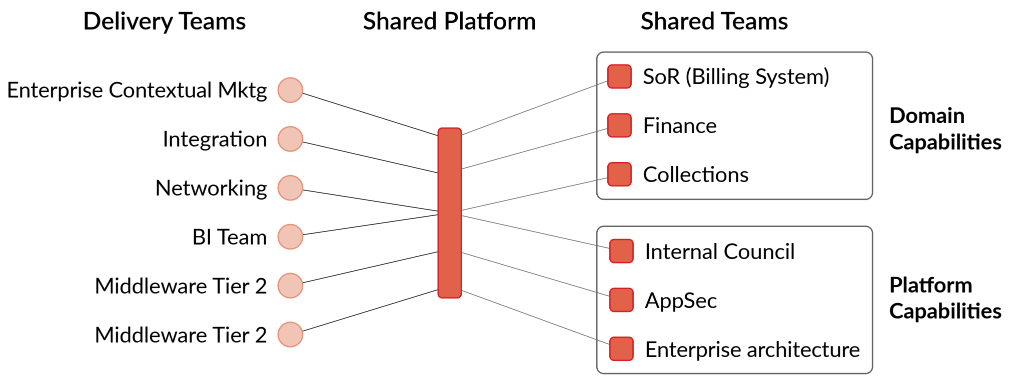


Figure 5: Shared Platform Example

Shared teams can be decoupled and decentralized by "productizing" their offerings in a shared platform.

Measure the Cost of Heroic Cross-Cutting Efforts

Cost is often a powerful motivator. It tells a story in language many leaders understand. Value stream mapping combined with financial data can reveal surprising insights. In the case of the Checkbox Project, it would have revealed in a couple of hours what lay ahead as

* Luca Galante, "What Is Platform Engineering?" PlatformEngineering.com (blog), accessed August 23, 2022, https://platformengineering.org/blog/what-is-platform-engineering.

† "Thinnest Viable Platform (TVP) Example Using a Data Platform," github.com, accessed August 23, 2023, https://github.com/sbalnojan/TVP-example.

‡ Satya Addagarla, Josh Atwell, Mik Kersten, Thomas Limoncelli, Sara Mazer, Steve Pereira, Jeff Tyree, Christina Yakomin, "Role of the Platform: Recommendations for Standardization," IT Revolution (blog), November 3, 2022. https://itrevolution.com/articles/role-of-the-platform-recommendations-for-standardization/.

months of struggle and uncertainty, prompting a careful and strategic approach. After the project, it would reveal the most costly and wasteful aspects of the initiative, which could be improved prior to the next effort.

Tactics to minimize the cost include the following:

- **Pick a product or initiative:**[*] Measure your most complete lead time and value stream metrics within that flow.
- **Find the constraint and contributing friction**:[†] Odds are the friction points and delays are linked to organizational dependencies external to the team involved.

Evolve Your Operating Model

Massive coordination costs, like those in the Checkbox Project, can be avoided by an internal operating model that resembles what most organizations offer beyond their boundaries. Most partner-driven initiatives are facilitated with various interfaces that enable decoupled action. The interaction typically follows an "If you do this, and we do this, we'll meet in the middle" framework. But as that dynamic evolves, it becomes more like a vending machine experience, with no direct interaction necessary.

Some options to evolve your operating model include the following:

- **Investigate Self Service:** Operational efficiency can be significantly improved through the customer-centric model of services and capabilities, enabled by self service.[‡] Published styles, operational models, and architectural guides serve as standard reference points, as do form, data, and SLA standards. When combined with API-first approaches, this means that resources and capabilities are available for consumption and contribution on-demand. This model encourages self-reliance and streamlines operational processes. Pulling just what's needed, when needed means less waste and more customer-supplier relationships throughout the organization, which translates to a natural extension to external customers and partners.

  Parts Unlimited could adopt a stream-aligned model (as in Figure 6), where

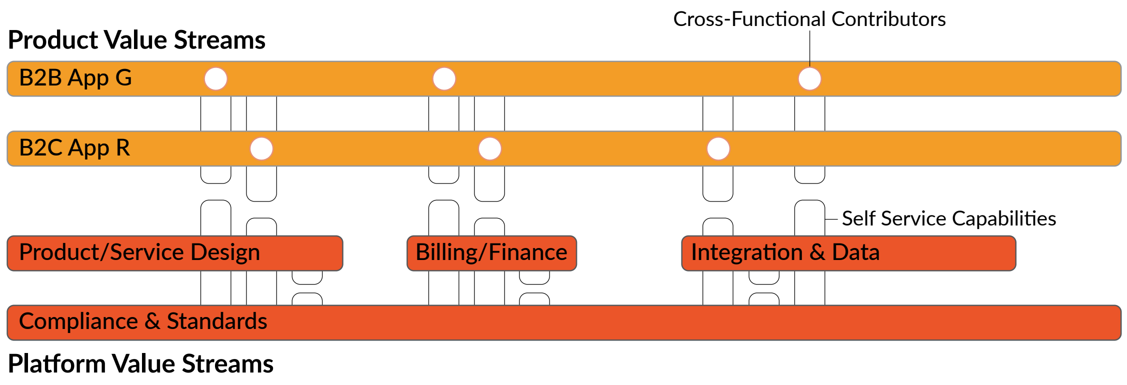
[*] Gene Kim, Jez Humble, John Willis, Patrick Debois, "Selecting which Value Stream to Start With," IT Revolution (blog), November 30, 2021, https://itrevolution.com/articles/starting-devops-value-stream/.

[†] Gene Kim, Jez Humble, John Willis, Patrick Debois, "Understanding the Work in Our Value Stream and Improving Flow," IT Revolution (blog), November 30, 2021, https://itrevolution.com/articles/improve-flow-devops-value-stream/.

[‡] Satya Addagarla, Josh Atwell, Mik Kersten, Thomas Limoncelli, Sara Mazer, Steve Pereira, Jeff Tyree, Christina Yakomin, "Role of the Platform: Recommendations for Standardization," IT Revolution (blog), November 3, 2022.https://itrevolution.com/articles/role-of-the-platform-developer-experience

each customer need is fulfilled by a stream-aligned team of cross-functional contributors able to access and leverage shared capabilities offered by internal platform streams. This not only reduces disruption and tight-coupling of dependencies, but creates a powerful "customer-centric" dynamic throughout the entire organization.

Figure 6: Example Self Service Model



Loosely coupled product and platform streams interacting via self-service enables flow

- **Investigate the Internal Partner Network Model:** At the advanced level, your organization can be reimagined as an internal partner network. By leveraging the experiences, models, and architectures of external partner networks, you can create a self-service access system to various capabilities. Salesforce's AppExchange marketplace,* for instance, is a great model to reference. Standard frameworks and SDKs, along with comprehensive documentation and support, form the backbone of this model. They provide their own apps, beta products, workflows, data, and thousands of partner apps on the system. It provides automated pipelines, low or zero-trust implementations, and automated verification systems. With this model, burden is matched with incentive, pushing individuals to fulfill all requirements before pushing for a change. This helps ensure operational efficiency and stability while encouraging personal responsibility and ownership.

Finally, regardless of where and how you start, you may benefit from familiarizing yourself with the principles in the next section. These transcend specific situations, methods,

* To learn more, check out the Salesforce *AppExchange 101: All You Need to Know* document here https://magicfuse.co/blog/salesforce-appexchange-101-all-you-need-to-know/

and guidance. They are applicable to every circumstance in which it is necessary to *organize for better outcomes*.

Supporting Principles

A common disease that afflicts management . . . the world over is the impression that "Our problems are different." They are different, to be sure, but the principles that will help to improve quality of product and of service are universal in nature.
— Dr. W. Edwards Deming

The Checkbox Project case study illustrates the challenges of implementing "apparently simple" new software functionality across diverse and heterogeneous teams, applications, and value streams. The authors' shared experiences are similar: the business feels frustrated about how it seemingly takes forever to implement an apparently simple thing.

IT is on the receiving end of this frustration. They experience equal frustrations themselves. The organization suffers, and mutual trust and respect is a common casualty. While everyone's "problems are different," and there is no one right way to organize for outcomes, we posit that a set of underlying principles could help software-dependent enterprises better organize to navigate these cross-cutting concerns.

The proposed principles[*] are organized into three subsets. One that reflects the basic people and organizational practices, one that helps us better treat projects and applications as products, and one that addresses underlying architectural factors.[†]

Organizational Principles

O1: Global Work Override

Global work will arrive unexpectedly and override local work.

Teams naturally optimize to best deliver the solutions or solution elements they are directly responsible for. Their backlog is full, and they are already operating at full capacity. They may not have visibility into upcoming global, portfolio, or significant cross-cutting work. When this work arrives unplanned, trouble follows. This may include additional work in

[*] Thanks to Alex Yakyma for peer reviewing the principles.

[†] Thanks to Don Reinertsen for providing the thought leadership and the pattern we are using to distill the essential and common truths that can be applied to address these problems. See Don Reinertsen's *Principles of Product Development Flow: Second Generation Lean Product Development* (Celeritas Publishing, 2009).

progress overload, which reduces overall team capacity for both types of work; missed commitments to a team's local customers; and a culture where teams seemingly appear unwilling to take on important cross-cutting global initiatives.

O2: Organizing Around Value Stream-Aligned Agile Teams

Cross-functional, value stream-aligned agile teams are the most effective and efficient value delivery mechanism.

Agile is a proven, effective approach to software and system development. Agile teams are small, cross-functional, cohesive groups with the skills and resources to define, build, and deliver value directly to customers with minimal handoffs and dependencies. Value delivered may be in the form of platforms (see principle A3 later in this paper), specialty components, or subsystems. Most preferably, teams are "value stream aligned" to be able to deliver full products and solutions to customers. This organizational model scales to teams of agile teams who collaborate in delivering larger solutions.

O3: Cross-Domain Planning

There must be a known way of planning for inevitable cross-cutting work, preferably on a cadence.

Teams of agile teams will rightfully optimize for local work, which is under their control and best serves their customers. That means it can be difficult for cross-cutting work to make its way into the system. In the case study, managing the new work was the responsibility of a centralized program management function.

However, such a model does not always engage the teams in innovating, planning, and committing to the new work. Nor can they be fully responsible for delivery (agile mantra: "the people who do the work plan the work they do"), as they are likely unaware of how their work affects the larger solution. Instead, enterprises should build a model where the teams participate in planning and committing to the cross-cutting work. Since the arrival of such work is not predictable, cross-domain and cross-team cadence-based planning addresses all local and global work as business as usual.

O4: Backlog and Road Map Visibility

Backlogs and road maps, including potential cross-cutting initiatives, must exist and be visible at all levels.

People do all the essential work of moving valuable work items through the system. But they can't possibly anticipate or advance work they cannot see. That means that all work in the system (current or upcoming global or local work, even work in process by other teams) must be made visible to all stakeholders. This can only be accomplished with a network of backlogs and road maps that are visible to all those involved. The resulting transparency also creates objective evidence of the system's state, enabling effective, real-time decision-making at all levels of the organization.

O5: Beware the Business Model Change

Initiatives that affect the business model will be many times harder and slower than those that don't.

As the case study indicates, cross-cutting initiatives may reflect a change to the underlying business model. When a business model change is needed, it will affect not just the systems under development but other extant and remote systems, multiple business functions, customers, suppliers, and partners. Ambiguous business requirements (see case study), alternative flows, and exception conditions may well rule the schedule (see also principle P3).

The impact is often significantly underestimated, resulting in delays and missed expectations. In this case, it is crucial to understand expectations of business outcomes for all parties. Implementing and integrating the change in vertical slices is prudent (see also principles A2 and A5). Building an MVP (minimum viable product) to test the new business model hypothesis is also warranted.

O6: Internal Customer-Producer

Effective internal customer-producer relationships help the enterprise self-organize to deliver cross-cutting initiatives.

Decentralization allows for decisions to be made by those who have direct contact with the problem. Fast feedback cycles are necessary to ensure that decisions are correct and any needed mid-course correction happens early and often. One way to establish this feedback loop is through the customer-producer relationship between and among teams.

These relationships reduce the need for top-down coordination as the outcome is achieved through a network of customers and producers who know how to bring the coordinated intent to life. Effective operation of the network requires tactical planning, synchronized execution, fast feedback, and alignment events that facilitate direct interactions between producer and consumer teams.

O7: Secure Capacity

Any realistic implementation plan for cross-cutting concerns must assure available capacity from affected parties.

Cross-cutting initiatives must compete for capacity with the team's local work and the commitments they have already made to their internal or external customers. Typically, their work is already planned to capacity. Every team involved must allocate some of their existing capacity to accomplish the new initiative. This often requires negotiation and tradeoffs as new objectives interfere with existing plans and commitments. Indeed, stakeholders cannot understand the full implications of the new global initiative until all the affected teams are identified, engaged, aligned, and able to commit capacity to the new work.

O8: Empowerment and Accountability

Empowering teams and elevating their responsibilities is of higher value than "order taking."

The success of each enterprise-wide, cross-cutting decision relies on countless local decisions that must be made quickly and in alignment with the overall mission. Organizations that rely solely on top-down coordination and centralized decision-making struggle to manage the complexities of cross-cutting initiatives. This is due to the challenge of understanding the day-to-day work required from people who may be several organizational levels away and because enterprise architectures require a significant amount of lateral interactions that are best handled directly by those involved. As such, leaders must focus on developing local expertise and competence and provide clear objectives and intent for new initiatives rather than providing predefined and overly detailed specifications.

Product Principles

P1: IT Product Management:

We must manage our applications and projects like products.

Traditionally, IT and software development work has been managed in the form of projects. A need was assessed, a project was defined, approved, and funded, and "resources" were gathered from various functional groups (development, test, project management, DBAs, etc.) to initiate and complete the work. The work was then handed to operations teams, where it was deployed, debugged, and mostly forgotten.

A more effective approach is to treat our significant applications like products. Products are long-lived and have customers (real people, either internal or external to the enterprise) whose needs can be understood, empathized with, and addressed. But products can't speak for themselves and require constant attention to ensure they continue to meet the customers' needs or are decommissioned when economics dictate. That means that individuals on tech teams need the purpose, skills, and attributes necessary to take on the role of Product Management (or Product Owners), thereby providing the focus required to ensure that the solutions serve their customers now and in the future.

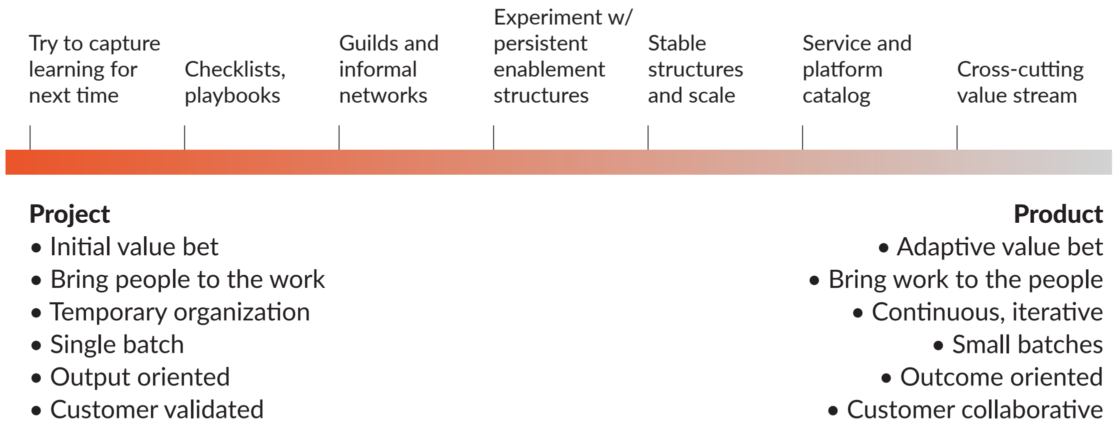

Try to capture learning for next time
Checklists, playbooks
Guilds and informal networks
Experiment w/ persistent enablement structures
Stable structures and scale
Service and platform catalog
Cross-cutting value stream
Project
• Initial value bet
• Bring people to the work
• Temporary organization
• Single batch
• Output oriented
• Customer validated
Product
• Adaptive value bet
• Bring work to the people
• Continuous, iterative
• Small batches
• Outcome oriented
• Customer collaborative

Figure 7: Project to Product Spectrum

P2: Cross-Cutting Learning Cycle

Implement cross-cutting initiatives via thin, integrated, end-to-end vertical slices.

Cross-cutting initiatives are complex and involve many unknowns, which must be revealed as early as possible. However, many organizations split the initiative into work packages that align with the organizational units responsible for the affected systems and subsystems. Consequently, each unit implements its segment, potentially iterating and learning, but the project is still "waterfalled" at the top level, leading to a delay in critical learning. To avoid this, organizations should stream-align around the new potential value and define and implement whole slices of cross-cutting increments of functionality.

P3: Underestimating Alternate and Exception Use Case Scenarios

Developers and business people routinely underestimate alternate flows and exception conditions, especially when the business model changes.

Alternate flows (different paths through the system) occur as the user learns the system or takes shortcuts when they become more sophisticated in using the functionality. Exception conditions happen when the user cannot achieve the goal for some reason. The long list of questions described in the Project Checkbox case study section on experience iterations illustrates the problem. Many of these questions are fundamental business model questions that can't be answered directly by the development teams.

Others touch on areas that may expose architectural problems. Answering those questions is a labor-intensive and calendar-consuming effort. Moreover, implementing and validating the functionality associated with these alternate flows may take far more time than the basic functional use case. Business people and developers must take this into consideration when developing the business case for the initiative. Otherwise, it will surely take twice as long as planned.

P4: Treat Platforms as Products

Stream-aligned teams are the customer of the platform product.

Enterprise architecture often involves creating platforms (see principle A3), which bundle related capabilities and functions into a cohesive asset with well-defined programming interfaces. These platforms solve problems like cognitive overload for the teams and code duplication. They typically have multiple internal "customers" who use the platform's capabilities to deliver business solutions. The success of cross-cutting initiatives depends on the effective management of these platform customer-producer relationships and the prioritization of platform capabilities. This requires treating the "platform as a product" and recognizing that "Yes, those developers— even the new and junior ones—are indeed the customer for our platform, and we need to treat them as such."

Architecture Principles

A1: The API Imperative

Architectures that expose functionality via APIs facilitate fast implementation of unanticipated new capabilities.

When there is no general approach to defining interfaces between different system parts, the code inevitably evolves in the direction of high coupling. This increases the complexity of the codebase and makes it increasingly more difficult to add new features. In contrast, using an Application Programming Interface (API) approach combats excess complexity by applying technology-enforced interface definitions and communication protocols. This results in a more modular, scalable, and flexible architecture that helps the organization respond to change and follow a more predictable and sustainable trajectory.

A2: Integration

If you can't integrate end-to-end functionality frequently, you will experience a high cost of delay.

When an initiative cuts across multiple teams and even organizational units, it is common for teams to focus more on implementing "their part" than ensuring that the functionality they create will work with the other pieces. As a result, serious issues add up between the parts, leading to significant rework or failure. A disciplined approach is required to integrate the different elements into a working increment frequently. A clear process, alignment, and technological enablement are required to support such integration. Many barriers may be encountered on the way to establishing frequent end-to-end integration (varying from a lack of proper integration environments to simple overconfidence that it will work). Teams must address these barriers systematically.

A3: Platforms

Digital platforms limit teams' cognitive load and coordination costs and facilitate cross-cutting functionality.

Enterprise application ecosystems quickly become very complex. Teams that are chronically operating under time pressure add new capabilities as fast as possible. As a result, the ecosystem becomes a "soup" of capabilities. A better approach lies in grouping the capabilities that serve the same purpose into a platform. This limits the cognitive load on stream-aligned teams (see principle O2).

For example, teams then know that there are platforms for user access, security, identity management, data analysis, business intelligence, compliance, and so on. These platforms needn't be defined and designed up front. The need will naturally emerge as similarly-purposed functions and components arise that can benefit from cohesive grouping and reuse.

A4: Cross-Cutting Features Stress Your Architecture

Your architecture is likely not up to the cross-cutting feature challenge; refactoring should be anticipated.

Most enterprise solutions are designed to perform a specific function, and everything in that solution is subservient to that responsibility. However, when a need for a new capability arises that cuts across multiple solutions, this new purpose is often incongruent with the functions each solution was initially built for. This discrepancy means that the architecture of each solution is not well-suited for unanticipated cross-cutting work. Therefore, it is simply a fact that large-footprint, cross-cutting initiatives will require architectural change—sometimes significant.

However, tools and methods can make this easier. Some architectures (API-based, platform-enabled, etc.) are more malleable, thus more supportive of new and cross-cutting work. Test automation and cross-solution integration mechanisms make large-scale refactoring faster and more predictable.

A5: Spike First

First, research the best course of action using small, dedicated teams of business and technology people.

Large-scale, cross-cutting initiatives often take a long time to implement. Even defining and understanding the immediate implications can consume much of the scheduled time. Expertise is required from many different areas, and the subject matter experts typically live in far corners of the enterprise. Because of its novelty, a cross-cutting initiative exposes impediments in a highly functionally oriented and siloed organization. Even after stream-aligning, silos may still appear due to the initiative cutting across established value streams. One way to quickly advance the organization's understanding of the initiative is to form a small, cross-discipline business and technology (enabling*) team from the constituent value streams to understand what the initiative entails. Such a team may even continue throughout the implementation, helping to align and coordinate the effort.

Conclusion

The Checkbox Project exemplifies significant pain points for large-scale, project-based organizational structure, communication pathways, and value delivery. It has exposed the hidden costs and limitations of traditional reporting hierarchies, functional orientation, and project-based disruption, often leading to a meandering and ineffective path towards customer value. The lessons derived from this study offer pathways for improving organizational agility and efficiency:

* "The Four Team Types from Team Topologies," IT Revolution (blog), February 28, 2023, https://itrevolution.com/articles/four-team-types/

- **Fostering Shared Visibility and Clarity:** By embracing visualization and mapping value streams, organizations can enhance collaboration and shared understanding. Even abstract simulation can reveal what painful experience may uncover in the future.
- **Optimizing Team Structure:** By leveraging capabilities within value streams, collaboration can be streamlined, improving time to market.
- **Architecting for Agility:** Decentralized and decoupled system architecture, domain-driven design, and embracing serverless and event-driven architecture options foster more agile development.
- **Measuring Costs:** Quantifying the expenses of cross-cutting efforts allows for a data-driven approach to minimize costs and improve efficiency. Not only capital costs but employee experience, opportunity costs, and beyond.
- **Evolving the Operating Model:** Investigating self-service and internal partner network models enable a more customer-centric and efficient operational process.

The accompanying principles described can be applied universally to help enterprises better prepare for the significant challenges that are common in projects of this type.

The Checkbox Project is a powerful testament to the need for organizational evolution toward better outcomes. Through the application of these strategies and the highlighted principles, companies can break free from restrictive silos and traditional limitations, aligning structure with customer needs, and enabling a quicker and more effective delivery of value. Future work will only further refine and enhance these principles, opening new doors for innovation and excellence in organizational design.

Transparency Statement for The DevOps Enterprise Journal

Journal Ownership: The DevOps Enterprise Journal is owned and published by IT Revolution Press, LLC.

Selection Committee: The journal is curated by Gene Kim, Jeff Gallimore, DevOps Enterprise Forum attendees, and DevOps Enterprise Summit Programming Committee in collaboration with the IT Revolution Press editorial team.

Curation Process: All article topics undergo an initial assessment by Gene Kim and other members of the Selection Committee respective to the specific issue. If topics are considered suitable, articles are then written by technology industry experts selected by the Selection Committee and or the IT Revolution editorial team. The final responsibility for editorial decisions rests with the editorial team for the journal. The journal does not accept unsolicited submissions.

Editorial Team/Contact Information: The editorial team can be contacted via info@itrevolution.com.

Copyright: All articles in the journal are published Open Access under a Creative Commons Attribution license (CC BY-4.0). Copyright is held by IT Revolution Press, LLC, and various authors. This allows the integrity of the work to be protected by IT Revolution while interested community members share, use, and build upon the work created. If used, appropriate attribution must be given.

Author Fees: All costs associated with publishing are held by IT Revolution Press. There are no fees for the author(s).

Conflicts of Interest: Financial or ethical conflicts of interest are assessed by the IT Revolution Press editorial team and clear statements of potential conflicts are given at the end of each article.

Frequency: The journal publishes two issues a year.

Access: All journal articles are published as Open Access at itrevolution.com.

Revenue Sources: All costs associated with publishing are funded by sponsors. Any articles specific to the sponsor are clearly delineated. The sponsors do not influence or choose any articles in the journal.

Advertising: The journal does not currently accept direct advertising.

Archiving: IT Revolution provides perpetual access to all journal content at itrevolution.com**.**

Direct marketing: On occasion, the journal will use direct marketing activities to raise awareness of the journal and to invite authors to submit articles. Sponsors may contact subscribers and readers who download the articles. Marketing activities are conducted by IT Revolution Press and sponsors.