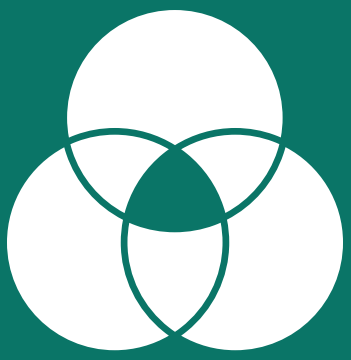
Moving from Project to Product



Modernizing Traditional Enterprise Operating Models

IT REVOLUTION
DEVOPS ENTERPRISE FORUM
2018

25 NW 23rd Pl
Suite 6314
Portland, OR 97210

Moving from Project to Product:
Modernizing Traditional Enterprise Operating Models

Produced in the United States of America

Cover design and interior by Devon Smith

For further information about IT Revolution, these and other publications, special discounts for bulk book purchases, or for information on booking authors for an event, please visit our website at ITRevolution.com.

PREFACE

In March of this year, we at IT Revolution once again had the pleasure of hosting leaders and experts from across the technology community at the DevOps Enterprise Forum in Portland, Oregon. The Forum's ongoing goal is to create written guidance to overcome the top obstacles facing the DevOps enterprise community.

Over the years, there has been a broad set of topics covered at the Forum, including organizational culture and change management, architecture and technical practices, metrics, integrating and achieving information security and compliance objectives, creating business cases for automated testing, organizational design, and many more. As in years past, this year's topics are relevant to the changing business dynamics we see happening across all industries and the role technology has to play within those changes.

At the Forum, as in previous years, participants self-organized into teams, working on topics that interested them. Each team narrowed their topics so that they could have a "nearly shippable" artifact by the end of the second day. Watching these teams collaborate and create their artifacts was truly amazing, and those artifacts became the core of the Forum papers you see here.

After the Forum concluded, the groups spent the next eight weeks working together to complete and refine the work they started together. The results can be found in this year's collection of Forum papers.

A special thanks goes to Jeff Gallimore, our co-host and partner and co-founder at Excella, for helping create a structure for the two days to help everyone stay focused and productive.

IT Revolution is proud to share the outcomes of the hard work, dedication, and collaboration of the amazing group of people from the 2018 DevOps Enterprise Forum. Our hope is that through these papers you will gain valuable insight into DevOps as a practice.

—Gene Kim
June 2018
Portland, Oregon

We've done Agile, we've done DevOps! Why do we still suck?

Introduction

Enterprises have invested significantly in modern software delivery methodologies—Agile, Scrum, DevOps, SRE, and cloud—to mobilize and accelerate their technology transformations. Yet many organizations still struggle to see the intended acceleration and, more importantly, positive outcomes from this investment. Why is this?

Their disappointment is mostly because the value of the new methods is diluted when executive management relies on old business management approaches to compete in the modern world. For instance, it is counterproductive to insert DevOps into a traditional project-based waterfall model. DevOps enables fast feedback loops and learning, but those are wasted if the technologists are learning on a weekly cycle while the business adapts on an annual one.

This guidance argues that there needs to be a fundamental shift from a project to product mentality, leveraging new insight derived by looking at the amazing transformation in the automobile sector. Consider the contrast between modern car manufacturing and IT. Car manufacturers invest in value streams aligned to their products and markets and optimize flow through those value streams. If car plants ran like we run IT today, we would allocate workers to five different lines concurrently, tear down the assembly lines once a year, and then shift the budget to the next project before the plant operations team had a chance to complete the automations they had invested the better part of the year in. It's hard to imagine how any meaningful volume of cars would get built this way. Yet this is exactly how the project-based mindset plays out in IT. We need to find a better way, and this paper proposes directional guidance to that better way in a shift from project orientation to product orientation.

This guidance also explains why the old winds of change are becoming new cyclones to make the case for change more compelling than it already was.

"Change or die" has never been more accurate for the leaders of our most important enterprises.

The Case for Change

Even though we are eighteen years into the twenty-first century, many large enterprises are only now seeing their fortresses buffeted by the swirling winds of *fin de siècle*. Where are these winds coming from, why are they coming now, and, more importantly, what needs to be done in their wake?

The Existing Winds of Change

The connected economy has meant that new "sharing economy" models have emerged to challenge existing enterprises. The original nature of Airbnb, Uber, and similar companies has morphed in the past three years from "sharing" to "monetized renting," and with this change, new extensive enterprises are emerging. New "cloud native" enterprises are often devoid of capital expenditure, have Agile development processes to accelerate new offerings, and can pivot business models (through acquisition) with agility and create robust cash flows. This development threatens the traditional enterprise because the old advantages inherent in scale (like huge warehouses, proprietary logistics lines, and access to capital) become defunct. In many cases, the economies of scale have become the burden of the enterprise. As a result, many large enterprises find that their primary infrastructure is now too expensive to support. This insight has led to a massive focus on cost reduction. In fact, a lot of new technology practices like Agile, DevOps, and the public cloud have been pursued precisely to right-size the economics of the traditional enterprise.

The New Cyclones

Two new regulatory weather systems are joining the already swirling winds buffeting the organization. The only way to address them is by shifting how we manage IT.

International Financial Reporting Standards

In the old world, senior executives could and did disguise their technology failings and boost the value of their shares by outsourcing their old technology. This strategy was widely adopted during the past twenty years. It provided short-term cost savings for the enterprise that were banked to show the illusion of efficiency. It created hope that somehow the sourcing parties could transform what the enterprise could not, and it shortened the balance sheet by removing assets that made everything appear better than it was. This retreat into concealing the true cost of IT and software delivery is at odds with the new International Financial Reporting Standards (IFRS).

IFRS 16 marks the end of off-balance sheet treatment of operating leases by lessees. Starting in 2019, enterprises will be required to recognize all the costs of leases associated with outsourcing as the recognition of a leased asset (i.e., right-of-use asset) and a corresponding financial liability representing its obligation to make future lease payments. The long-term adverse effects of the outsourcing strategy will come home to roost everywhere, and the costs will find their way back onto the balance sheet. This will certainly affect the current on/offshore mix of work, as well as the large-scale outsourcing of services, making it more important than ever for value from IT to become more visible.

General Data Protection Regulation

The European Union has introduced privacy regulations that will expose executives to potentially excruciating pain. The General Data Protection Regulation (GDPR) is a regulatory game changer that has a sting in its tail much more onerous than the compliance requirements themselves.

The cumulation of investigations, notifications, reverse burden of evidence, regulatory fines, vicarious liability, class-action suits, and individual executive exposure to prosecution make GDPR the mother of all regulatory regimes. But it is not only European firms that face this financial exposure; 80% of US firms will fall within GDPR. It is not just consumer data, either; it is also employee data. GDPR is about citizens' data in all forms.

Ninety-nine articles make up GDPR. Many of these can have far-reaching implications for large organizations, both public and private. In particular, we can expect GDPR to have three significant effects on the way large organizations view technology.

First, GDPR Article 25 refers to security by design. This will undoubtedly accelerate the implementation of DevSecOps. Second, articles such as 17, which refers to the right of erasure or "right to be forgotten," and 20, which covers data portability, are expected to accelerate the shift from application-centric IT to data-centric IT. Third, Article 25 refers to data protection by design and by default. This is one of a series of articles that regulate the relationship between outsourcers and clients. This is expected to influence the way in which testing is done in a DevOps environment, or indeed the way in which any form of data is moved and stored between an outsourcer and the client.

Given the enormous consequences of getting it wrong, the enterprise needs to focus as much on mitigating exposure as on ensuring compliance. The new focus of an enterprise will be to create certainty around the processes for creating, storing, analyzing, managing, and reporting data. The functionality of applications will become as or less important than managing the potential exposure to GDPR.

In these three situations, the move from a focus on projects to a focus on products will strengthen the holistic management of both security and data, making it less fragmented and offering a greater sense of control and continuity.

Change, Decline, or Die?

Tinkering with methods and techniques is no longer enough. This guidance brings together the threats, new insights, and potential of investments already made in exciting technologies to create a single new mantra as the basis for your route to value.

These disruptions are becoming increasingly common, forcing business and IT leaders to strategize, battle, and react with a more introspective approach.

As business leaders on either side of the technology paradigm, the common issues faced by most organizations stem from misalignment, lack of visibility, and the difficulty of defining and meeting expectations.

For the reasons listed in this paper, business and IT leaders must take a proactive approach to understanding the negative internal forces that are interfering with their ability to react to these disruptions. To help leaders understand the negative internal forces, this paper outlines the common issues that organizations are facing and provides guidance to help business and IT leaders navigate and tackle these issues.

The Five Problems

Problem 1: IT Is Disconnected from the Business and the Vision

The relationship between IT and the business is often a customer-centric view, meaning that IT views the business as a customer and that neither side views the other as a partner. The fact that IT professionals typically refer to their business partners as "The Business" is a simple example of this. This relationship fosters a feeling of taking orders or just delivering what was asked for instead of being a trusted advisor and working through the value proposition together. "The customer is always right" never works in the context of building business value. In many cases, IT lacks a clear definition or understanding of the customer, whether internal or external.

In the project (scope) style of delivery, the business gathers a large list of requirements for IT to execute. IT returns an estimate of what it will cost to implement the requirements. The business responds with what they are willing to spend. This paradigm of delivery often leaves IT disconnected from the true purpose of the business vision and wondering why they are working on specific requirements instead of focusing on delivering the value.

Problem 2: Business Feels that IT is Solving Its Own Problems Not Delivering More Value

Not unlike the previous internal force, this negative force is the business leaders' perception that IT is just solving their own internal problems, not delivering value to the business. This directly contributes to IT being perceived as a cost center as opposed to a profit center. If the business sees the work being delivered by IT as directly related to the bottom line, not just in cost but also in value to the customer, IT will start to become a trusted partner, not just the technology problem solver.

Problem 3: Leadership is Tracking the Activities Not the Results

Due to the historically negative results from project-based delivery, there has been a increasing lack of trust between the business and the project development teams. This has caused project managers in IT departments to focus on processes that track and report on project activities to ensure that costs are kept in check and that conflicts

and risks can be reported to executives. However, the activities of project delivery often do not create a clear representation of the outcome of those activities.

By tracking the outcome or results of the project instead of the activities, a true representation of the entire value stream is visible, not just the technology deliverables. This concept is key to the migration from project to product. Product-based teams build and measure outcomes; they don't focus on the activities of the delivery process. Those outcomes drive refinement of processes to increase the value of the outcomes.

Problem 4: Project Funding is Fundamentally Broken

The recurring theme of each of these negative internal factors is that project-based delivery is inherently broken, and at the core of project-based delivery is the funding model. Because funding projects (scope) takes all the requirements the business is asking for and then estimates by delivery organizations, the value of those requirements is lost within the concern over delivery of those requirements. The mantra "on time, on scope, on budget" resonates throughout every traditional project management office, so breaking away from the traditional project model means changing our financial focus to value outcome over being on budget. In a product-funded model, the business has a budget for product development or enhancement. Feature requests are entered into a backlog and prioritized by the product owner based on value. This assures that the most value will be delivered for any given budget.

Problem 5: IT Feels Like a Black Box to the Business

Nearly every organization struggles with the communication of IT delivery processes to their business counterparts. This communication typically takes the shape of status reports or status meetings to help the business understand the complications of a project's current state. The formulation of these reports and statuses are IT's attempt to pull the business closer to their work and build the partnership that is inherently missing. The business leaders, on the other hand, often struggle with this style of communication because the issues IT delivery teams face are typically issues around the team's struggles to deliver due to scope, time, or budget, not a message about the impact to the business outcomes.

As hard as organizations try to create alignment with the business in the project-based approach to delivery, the difficulty is creating the right level of visibility and then presenting the visibility in a consumable manner that both the business and IT can relate to. Data-backed visibility and transparency will help to instill the trust needed for a true partnership between IT and the business.

All these negative forces are feelings and realities that have been persistent hurdles to achieving a nimble, proactive organization capable of not only combatting disruption but also being the disruptor in a given industry.

Recommendations for Moving to a Product-Based Model

So, now that you want to move from a project-based to a product-based model, how do you start? In a large enterprise, there is likely an embedded culture of working in a project model. Roles such as project managers and portfolio managers will be impacted. Business leaders who previously may not have directly concerned themselves with IT's implementation of Agile will now need to be more involved. We will first clarify what a product-based software delivery model is, then provide some simple guidance on how to start your journey.

Product-Based Software Delivery Model

When trying to appreciate the idea of "product" within the landscape of IT and DevOps, it can sometimes be difficult to understand the boundaries of a product team in relation to how our business counterparts define a product. To help conceptualize the idea of IT product teams to market products, we'll lean on the product life cycle (PLC) as it defines products in the marketplace. The PLC is a simple model for describing the life of a product in a market using the relationship of measurement between costs/time and subsequent sales.

Every product sold within the consumer world today has a life cycle defined by the creation to the demise of that product. Although every product has unique factors that led to its inception and rise in maturity, the fundamental model for understanding and recognizing those stages remains constant. A product's life cycle follows four stages:

- Introduction: The product is new to the market and hasn't had a lot of traction since being released.
- Growth: The product hits a tipping point at the end of the introductory stage that pushes it into a rapid growth mode.
- Maturity: At this point in the product's stage, it is firing on all cylinders, but the product begins to show a slowing of growth and eventually shows signs of decline.
- Decline: The product has started to show signs of weakness and is losing its value to customers.

Figure 1 shows the relationship of the life cycle stages to sales and time.

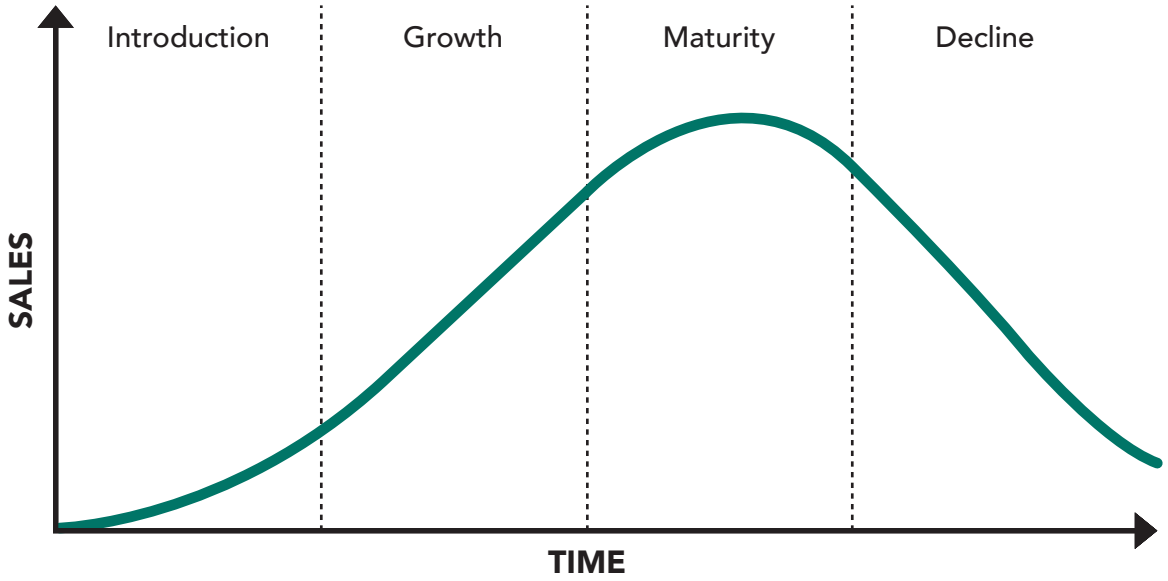


Figure 1: Product Life Cycle Stages

Now that we have a general understanding of PLC, there are several interesting relationships between the PLC and IT product teams worth pointing out, including the following:

In Figure 1, imagine that an IT product has been introduced to the ecosystem. Now, replace the label "Sales" on the y-axis with "Resources." As capabilities are introduced to customers (internal or external ecosystems), the resources needed to care and feed

for that capability rise to a point of diminishing returns. In this example, resources could be capital investment dollars, manpower, and/or CPU/network/storage usage.

The most obvious abnormality to how the two ideas contrast is that software products don't magically appear. This model doesn't consider the work needed to "introduce" the product to the market. In our guidance, we strongly suggest introducing "product" teams to alleviate many of the pitfalls common in today's corporate environments. However, we also believe that projects don't necessarily go away.

In Figure 2, we've replaced "Sales" with "Resources" on the y-axis to better represent a software delivery point of view. We've also added the gray section entitled "Creation." We believe that this part of the life cycle could follow more of a project life cycle in the ideation and creation of the initial product.

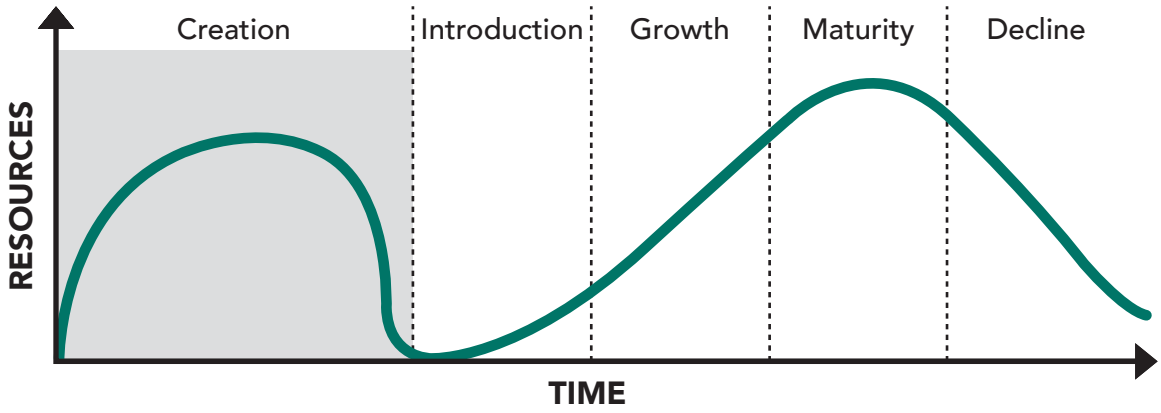


Figure 2: Software Product Life Cycle

The reason for the contrast in delivery models between projects and products in software is the introduction of the idea of a minimally viable product (MVP). For a product to reach the point of viability, a specific scope must be met to deliver that value to the customer. Imagine building a house. To get any real value from the house we need a few crucial features: a roof, flooring, running water, electricity, bathrooms, windows, doors, and a kitchen. What makes *this* house an MVP is that the builder, knowing he could sell the house with these minimal features, installed a standard tub, no dishwasher, cheap carpet, and garage doors with no opener.

The project portion of the house analogy is the idea of building a house, deciding on the smallest number of features we need to live in the house, and building the house. Once we assemble the house with our minimum set of features, the house

comes out of the project phase and enters the product phase. The product portion of the house analogy is identifying features that we need to make the house comfortable, less risky, and more usable based on our changing needs.

Now, let's apply that same thinking to software deliverables in general. To get value from a software platform, we need a base set of features that the customer will require to purchase or use the platform. This minimum set of features is our MVP. When new platforms or software are being dreamt up, the team will need to be assembled and introduced to the idea that will create the outcome. In many cases, this team won't exist, and one will need to be assembled. Once assembled, the team will take off on the defined scope to produce the MVP. The scope typically comes with a defined budget for how much the team thinks it will cost to deliver the MVP.

Following this approach, projects turn to products at the point where the team has delivered the first MVP of the product. Once the product is released to the customer and value has begun to be measured, the team will start to iterate on features to add additional value to customers or reduce risk as the product matures. This iterative approach requires an iterative mindset with a stream of funding to fund the capacity of the team to iterate.

But, as we've laid out in the PLC (Figure 1) and the software PLC (Figure 2), all products will decline in use and value. This decline has a direct impact on the resources used to curate the product, which translates to a reduction in those resources. As products reach the "Decline" stage, software product teams begin to lose the prioritization of features and consequently funding for the team.

From the inception of the idea through the creation and growth of the value to the ultimate reduction and elimination of that value, the software PLC shows a life cycle where the project creates a baseline in which the product begins to return value from the initial investment. The product then follows the PLC through iterations and investment to the demise of the product and the next idea. This relationship is the true project-to-product evolution.

Table 1 further clarifies the different approaches and associated tradeoffs.

Table 1: Differences Between Project- and Product-Oriented Approaches

	Project Oriented	Product Oriented
Budgeting	Funding of milestones predefined at project scoping. New discretionary budget means the creation of a new project.	Funding of value streams adjusted based on business results. New budget allocation based on demand.
Timeframes	Term of the project (e.g., one year). Defined end date. Not focused on the maintenance/health after the project ends.	Life cycle of the product (multiple years) includes ongoing health/maintenance activities.
Success	Cost center approach. Measured to being on time and on budget. Capitalization of development results in large projects.	Profit center approach. Measured in business objectives and outcomes met (e.g., revenue). Focus on incremental value delivery and regular checkpoints.
Teams	Bring people to the work. Allocated up-front, people often span multiple projects and exhibit frequent churn and reassignment.	Bring work to the people. Stable, incrementally adjusted cross-functional teams assigned to one value stream.
Culture	Often "top down"-driven, managed to deadlines, lack of decentralized decision making.	Empowered and highly collaborative organization willing to experiment and learn from failure.

Prioritization	Program and portfolio management, project plan-driven, with a focus on requirements delivery. Projects often drive waterfall orientation.	Roadmap and hypothesis testing-driven, with a focus on feature and business value delivery. Products drive Agile orientation.
Strengths	Resource allocation and planning for highly certain, low-variability work.	Fast feedback loops and learning; high-variability work.
Delivery	IT is a black box. Project management offices create complex mapping and obscurity.	Direct mapping to what the business wants that enables transparency.

Guidance to Get Started

So, where to begin? The following will provide some recommendations, but it should be noted that this is not something where "one size fits all." Every organization has its own culture and will take its own path to success. Consider the following as advice on things that have worked at other enterprises rather than an exact blueprint for your organization.

1. Priming the Pump

Before actually talking about moving to a product-based structure, there's some work that can be done to increase your chances of success. Consider moving from a model where people are moved from application areas to form project teams to a model that moves the project work to these established teams (see Figure 3).

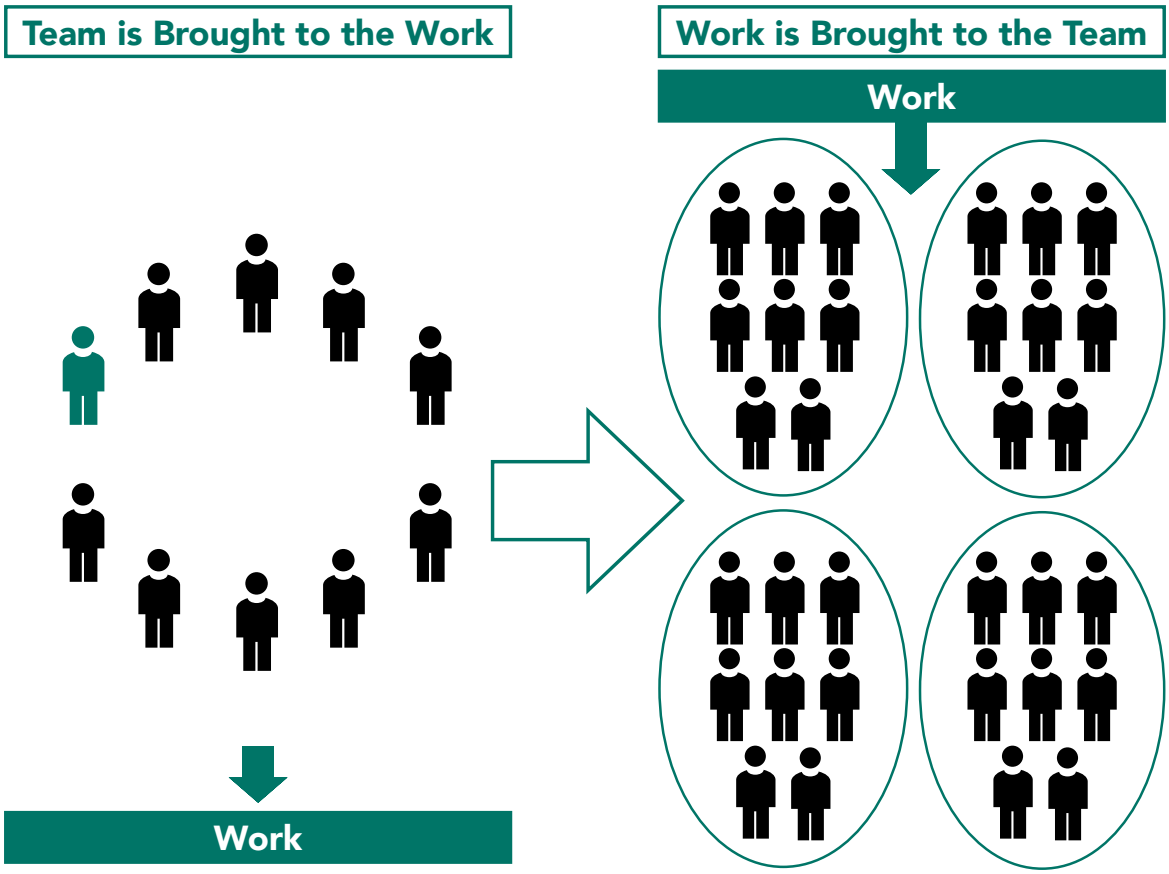


Figure 3: Team/Work Structure

This mitigates the issue of having to constantly form and reform teams around projects that are temporal in nature and don't provide a sustainable model of getting work done. So while there may still be portfolios of projects and project managers, the heavy lifting will be done by the established Agile teams that are impacted by the project. If a given feature of a project has stories that have to be completed (e.g., Story 1 by Team A and Story 2 by Team B), this work will flow into the backlog of those teams who will pull that work in based on priority.

Ultimately, you would look for this priority to be determined by the product owner. However, at this stage in your journey, you may look to this role being filled by internal IT personnel. It's important to note in this model that control for what work gets prioritized and delivered shifts from the project managers to the product owner (and broader Agile team).

2. Start Small

As with other Agile initiatives, it pays to start small and build momentum. Changing the culture of a large organization is not something that can be forced, even if there is top-down support. But many times, these types of initiatives that don't have full support in the beginning can be expanded, as is described in the 2017 DevOps Enterprise Forum paper *Expanding Pockets of Greatness*.[1]

Finding one area of the enterprise that is open to working in the product model is a great way to start. This can be used to demonstrate that the product-based model is possible to accomplish, and this area's success can be used to counter resistance. It changes the conversation from "This can't be done here" to "This is how it can be done here." The following are some criteria that can be leveraged to maximize the chance of success:

- **Have a supportive IT leader**. There needs to be someone at an executive or leadership level that is willing to be a champion for the team. This is a journey. Things will not always go smoothly, and the team needs cover from a strong leader who has their back.
- **Have a supportive business leader**. While many Agile or Lean efforts have traditionally been driven from inside IT only, this transformation can't be done without a willing business champion. This will require changes in how the business and IT work together. The business and IT leaders involved have to work as one team and be willing to address risks associated with this change while providing servant leadership by supporting the teams, listening to their needs, clearing impediments, and modeling the way for the overall change in behavior needed.
- **Start small**. Within the portfolio of that area, identify a product (or small set of products) for which an MVP can be defined and for which a single two-pizza team[2] can support.

---

[1] "Expanding Pockets of Greatness: Spreading DevOps Horizontally in Your Organization." IT Revolution. June 27, 2017. Accessed May 08, 2018. https://itrevolution.com/book/expanding-pockets-greatness/.

[2] Stack, Laura. "Jeff Bezos' Two-Pizza Rule for Building Productive Teams." *The Business Journals*. November 22, 2016. Accessed May 08, 2018. https://www.bizjournals.com/bizjournals/how-to/human-resources/2016/11/jeff-bezos-two-pizza-rule-for-building-productive.html.

- **Assign a product owner**. There needs to be an assigned product owner from the business who can sit with the team and manage the funding and prioritization of the work to flow it into their backlog. This requires a significant commitment from the business, as product ownership requires a much deeper and more ongoing business engagement than traditional project management models.
- **Reduce lead time by providing more autonomy**. In order to reduce lead time (improve flow), product teams need more autonomy to plan and deploy business capabilities across technology domains (e.g., web, mobile) and across impacted applications (perhaps using an innersourcing model[3]). Assuming the product team has the technical ability and expertise, they can make the changes to the application and issue a pull (or merge) request to that application owner. That owner can then review, validate, and accept the change, including it into their application, which can then go into their next application release.

3. Label This an Experiment

Even with IT and business leader support, this should be viewed by the enterprise as an experiment. It's an attempt to learn what works, what doesn't, and how the organization can be improved.

In order to run an experiment, there must be criteria to determine if it is, in fact, successful or not. Examples of metrics to track are things like flow metrics[4] (e.g., flow of business value stories). The goal of moving to the product model is to increase flow and reduce lead time while maintaining or improving quality and cost. In reality, there is data that demonstrates quality and cost will be improved, but at this point, don't set the bar too high. Simply to "do no harm" will suffice if flow is improved.

There is also an aspect of this relating to W. Edward Deming's concept of "quality circles."[5] The team itself understands best what is inhibiting their flow and how to improve it. The team should use retrospectives to ask themselves why certain stories had shorter lead times than others and what improvements can be made. If there is a

---

[3] "Innersourcing." Innersourcing. Accessed May 08, 2018. http://www.inner-sourcing.com/.

[4] "Value Stream Architecture." IT Revolution. July 12, 2017. Accessed May 08, 2018. https://itrevolution.com/book/value-stream-architecture/.

[5] Deming, W. Edwards. *Out of the Crisis*. Cambridge, MA: The MIT Press, 2000. 22.

need for a change that extends beyond this team, it can be escalated up through leadership to address the application of Deming's concept of "systems thinking."[6]

4. Make the Work Necessary to Adopt This Model Visible

After establishing the initial experiments, there needs to be a focus on continuous improvement to continue adapting this model. Per Dominica DeGrandis's book *Making Work Visible*,[7] make this continuous improvement work visible. Leveraging kanban boards and placing your work in public viewing areas, like on TV monitors or walls near where your teams sit, is a great example of this. It will take team bandwidth to adapt this model, so be transparent about it. Track blockers just like the team would for any other business deliverable and escalate those blockers to their leaders to address.

5. Have Show and Tells and Invite Executives

One powerful way to move culture is to have retrospectives and show and tells (at least once a month) that focus on the adoption of this model. Having executives share the energy and the excitement of the team as well as the successes and areas where they need help can be a very powerful way to impact the executive mindset, which is necessary to expand the product model to other areas. Giving executives a sense of ownership over supporting the team in making this work will pave the way for future success.

6. Celebrate Small Wins

Many organizations are not good at celebrating and publicizing wins. We talk a lot about what is going wrong, but we don't talk as much about what is working well. Stories are powerful! People remember stories being told by the teams doing the work. When other teams see their peers succeeding, it is powerful proof that this is not just a theory but a possibility right here, right now. Having the business leader talk about how this model is helping them succeed in the marketplace is extremely effective.

---

[6] DeArdo, Carmen. "Systems Thinking." LinkedIn. February 02, 2017. Accessed May 08, 2018. https://www.linkedin.com/pulse/systems-thinking-carmen-deardo/.

[7] DeGrandis, Dominica. *Making Work Visible: Exposing Time Theft to Optimize Work & Flow*. Portland, OR: IT Revolution Press, 2017.

Having teams tell their stories and sharing credit for success is an important component to build the critical mass necessary for broader adoption.

7. Have the Patience to Persevere

The journey ahead will seem daunting. It is necessary to reflect on how far you've come and not just on how far there is to go. Reach out beyond your company to the broader DevOps community and to others who are on the same journey or who have already successfully demonstrated success in the product model. At times it may feel like a scene out of Dickens, being the best and worst of times.[8] Have patience, persevere, keep the faith, and don't be afraid to ask for help. It won't be easy, but it will be satisfying to see the progress that you will make.

Use Cases

The following use cases show examples of common challenges in software development organizations and how they play out differently in a product-oriented versus project-oriented culture.

Case 1: Priorities Coming from All Directions

A group of developers are very busy working on a project, and an email sent from their compliance team alerts them that their applications need to be upgraded to the latest Java version by the end of month. Almost simultaneously, another email rolls in from the security team that a vulnerability was also found that needs immediate attention.

Project Paradigm: Developers begin complaining to their project manager, who consequently calls a meeting with both security and IT, and raises concerns about potential project schedule delays. The developers ultimately get the message that they need to do everything. The project schedule slips, the other initiatives are partially completed until the project delay gets too much visibility, and the developers are told to shift back to project work. The developers are frustrated, and no one is happy about the outcome.

---

[8] DeArdo, Carmen. "From Dickens to DevOps." LinkedIn. March 25, 2018. Accessed May 08, 2018. https://www.linkedin.com/pulse/from-dickens-devops-carmen-deardo/.

Product Paradigm: The product owner gets word of the conflicting priorities. They immediately have conversations with security to assess the risk of the vulnerability. Next, they track down the IT platform owner to understand the impact of the technical debt of the Java version change.

After understanding the priorities from security and IT, they engage the business to determine the urgency and impact of taking longer to deliver the features at the top of the backlog. As the owner of the product backlog, they prioritize the stories appropriately and plan for the next sprint meeting. During the sprint planning session, the developers provide their feedback and ideas about how they can best address all of the new needs.

Case 2: Conflicting Initiatives

An initiative has been funded to add a new capability for the business, and a team is put together to plan this change. First, they figure out what software components need to be modified; then the development work begins. Around the same time, another initiative is born that will require modifications to some of the same software components.

Project Paradigm: The first initiative will spin up a project team and allocate resources. With some help from the business analysts, the developers get an idea of what might need to be changed, and they start creating backlog items. Sprint planning begins, and they try to plan for features that they can show or maybe even deploy to production incrementally. Then the new initiative comes along and a new product manager puts together a team. At some point, one of the developers on the first project notices that there are commits to their GitHub repository that they didn't make. They track down the developer and ask what they're doing in there. Words fly, and management is called in to work this out. IT goes back to the business requesters to tell them they can't do everything and asks them to prioritize what is more important. One of the initiatives ends up being put on hold while the other completes its work.

Product Paradigm: Since both initiatives touch the same product portfolio, the single product manager is aware of both. When the second initiative is brought up, the product manager gets a couple of business analysts who also worked on the first initiative to break it down and determine the overlap. They evaluate whether parts of each initiative can be met in parallel or if they need to be serialized. The intact development

team is also brought into the conversation to assist in evaluating. Ultimately, new stories for the second initiative are added to the backlog. For each sprint, the developers who are extremely familiar with the code make suggestions about order and common solutions for both initiatives when possible. With each sprint, they deliver new functionality to the products, which meets the needs of the business.

Conclusion

Enterprises can no longer rely on long-established business models in which technology was merely applied to drive more efficient business processes. In our current digital economy, technology is now key to all business strategies. The traditional approaches to manage technology planning and delivery are not optimized for this new reality. These models were optimized for cost and segmented responsibilities to drive localized efficiency of the different processes across the technology value stream. These models need to fundamentally change to drive more connectedness between business and IT. This enables businesses to shift focus from optimizing for cost and a false sense of timeline predictability to optimizing for the speed and rate of business value delivery. Those that have embraced this shift or were born in this era are disruptors that are uniquely positioned to capitalize in this new economy.

For the rest of you, being Agile and implementing "code to cloud" DevOps practices are necessary but no longer sufficient to keep competitors from disrupting your business and taking your customers! The differentiation has now shifted left in the value stream and requires a new way of thinking about how work flows, how to make bottlenecks visible, and how to empower product-based teams. Will this journey be easy? No. Is it necessary to start now? As DevOps inspiration W. Edwards Deming is often quoted as saying, "It is not necessary to change. Survival is not mandatory."

Contributors

- Ross Clanton, Technology Transformation, Verizon, @rossclanton
- Carmen DeArdo, DevOps Leader, Nationwide, @carmendeardo
- Mik Kersten, CEO, Tasktop, @mik_kersten
- Alan Nance, EVP, Leader of Digital Transformation Business, Virtual Clarity, @alan_nance
- Karen Person, Build & Release Engineer, Standard Insurance
- Jason Zubrick, Director of Architecture, GameStop