The FIUYMI Fallacy

January 17, 2022

Director, Community Ecosystem Engagement - Cybersecurity

Michael is a world-leading IT industry analyst. He has led North American and global initiatives focused on developing insights and strategies that connect technology solutions with business needs, combining data, knowledge, analysis and advanced content delivery to define options for IT and buy-side businesses.

Submitted by Michael O'Neil on 17, Jan 2022
The FIUYMI Fallacy

“Fake it until you make it.” Every person reading this blog has heard that phrase, most likely from a business unit manager who is willing to forego a robust development cycle in order to deploy a new capability or product as quickly as possible, intending to flesh out the offering once it achieves traction with customers.

Technical disciplines like software development don’t have a direct analog to FIUYMI, but the Agile/DevOps world[1] offers important parallels to both the pursuit of speed and the need to transition the approach to concentrate on quality. In Agile, rapid sprints allow for development of targeted capability before a functional product is produced; the MVP (minimum viable product) is not a ‘fake,’ but it’s not intended to be complete at time of release.

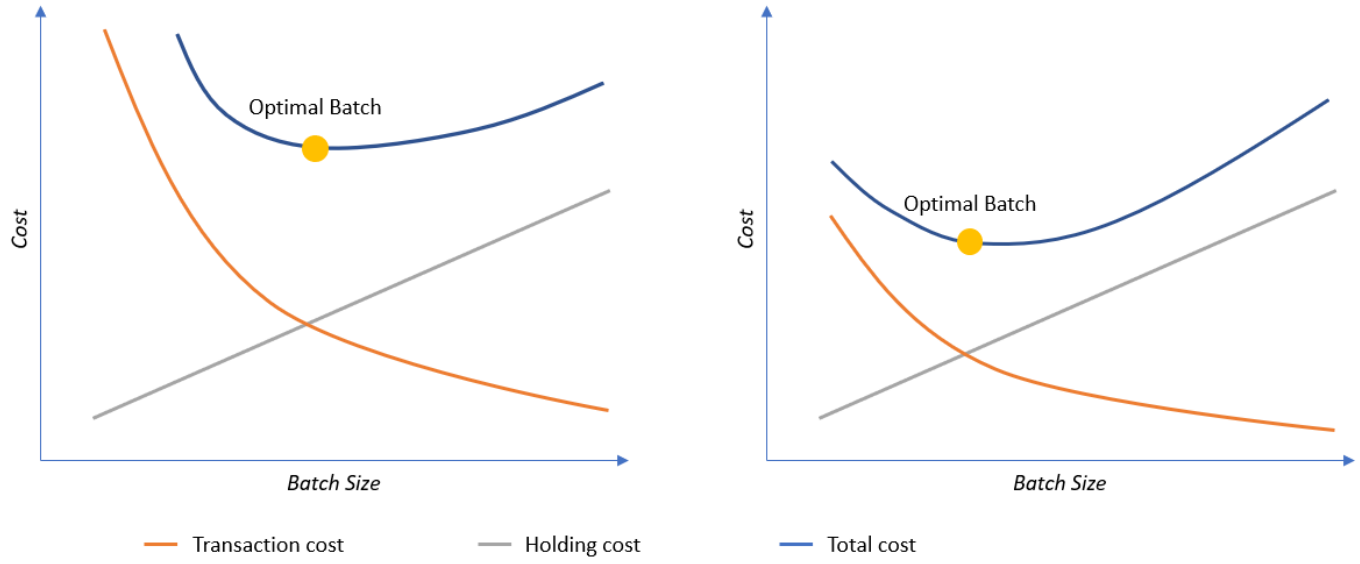
MVPs enable organizations to a stand-in for a more fully-functional successor product that is envisioned as a follow-on for the MVP if and as the associated business case is borne out.

In context, it’s important to see Agile/MVP as a means of optimizing portfolios, not individual products. An MVP approach allows an organization to assess multiple investment paths in parallel, giving business leaders an array of options for improving overall performance and returns.

In some settings, security and user experience are considered essential to an MVP – but this is not really intrinsic to the DevOps/Agile approach. In the landmark *DevOps for the Modern Enterprise*[2], Mirco Hering illustrates the relationship between "transaction costs" (set up costs), "holding costs" (defined by Hering as "the increasing cost of fixing a problem later in the life cycle and the missed benefit of features that are complete but not in production yet") on total costs and "optimal batch" sizes – the scope of the product produced. The graphic on the left shows how high transaction costs – which might be associated with legacy infrastructure and Waterfall development technologies – result in higher costs and larger batch sizes, and that DevOps enables lower-cost, more rapid releases of smaller batches of code.

The cost and agility benefits of the latter approach are clear in the diagram. But while holding costs – which in context, might well include security and user experience, both attributes that are more costly to address later in the lifecycle. Unless security and user experience are baked in 'by design,' minimally-viable products may lack features that are critical to optimizing the value of a specific software product, or which are costly to bring into line with best practice requirements. It might be what is viable for you to deliver given your constraints, but it may not provide enough value for users, or it might expose you to security risks that out-weigh the minimal benefits.

Relationship between transaction costs and batch size (Source: DevOps for the Modern Enterprise)



Reproduced from *DevOps for the Modern Enterprise*

This isn't (at all) an argument to return to the bygone days of waterfall development practices: with no slowdown in business demands for an ever-expanding digital business platform, reverting to a more deliberate and costly approach to software development isn't (and shouldn't be) an option. However, there are sound reasons for IT and business executives to navigate a path that addresses the need for both agility and software quality.

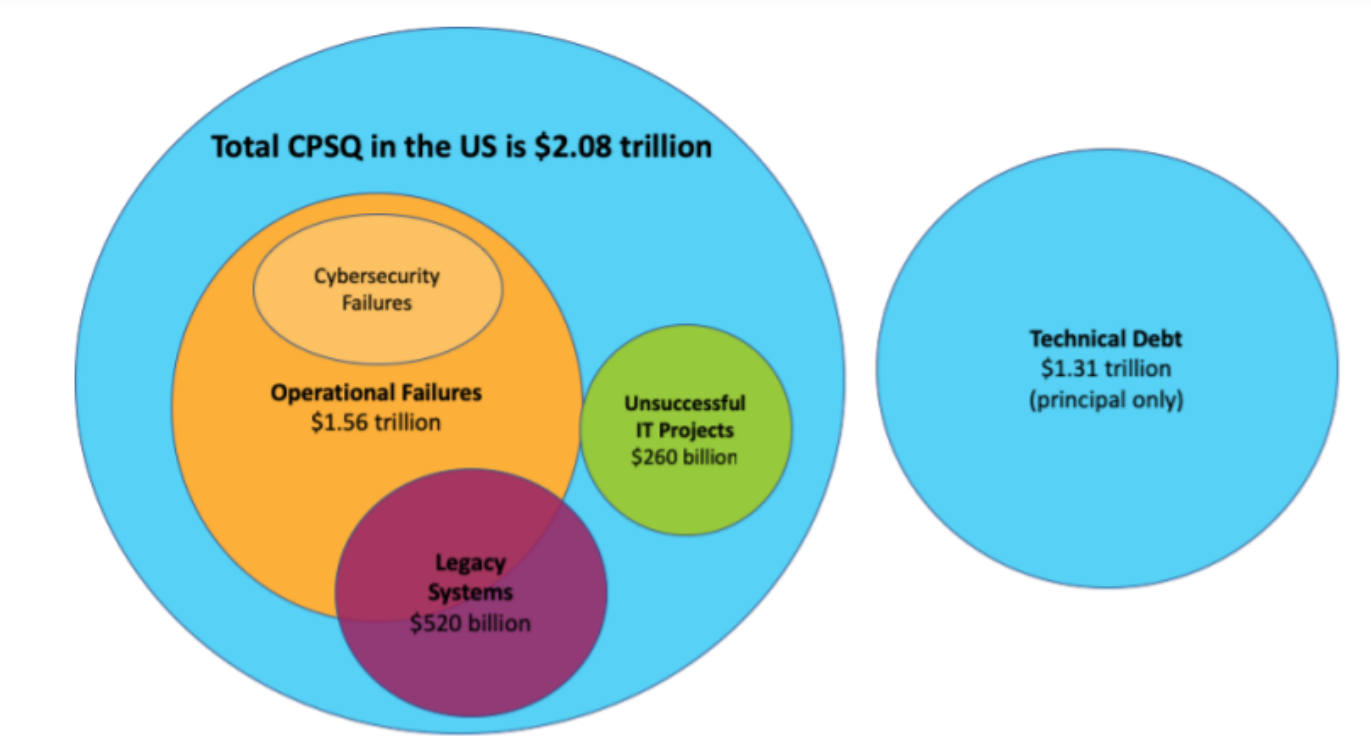
The $2T Question

Software quality is a major factor in financial viability across virtually every sector in the economy: software is the backbone of operations in government, the medium for exchange in financial services, the single most expensive component in new automobiles.

Despite this prominence, businesses pay an enormous toll for the "Cost of Poor Software Quality" (CPSQ)[3]. A report issued in January, 2021 by the Consortium for Information & Software Quality pegged the cost of CPSQ to US-based businesses at $2.08 trillion in 2020. The report holds that this figure may be understated, due to difficulties in establishing the true cost of cybersecurity failures, and adds that the $2.08 T figure does not include an additional estimated $1.31 T in technical debt, which is treated as a future cost.

Stratascale experts polled for this article added other quality dimensions to the mix: Lead Technical Advisor – Automation and Digital Experience Chris Hudson notes that "speed of adding/modifying features, lower operational costs and cloud-native solutions that are able to scale up and down on demand are other indicators" of quality, while Head of Innovation Labs Research Derek Shank observed that "in the same way that health is much more than the absence of disease, quality is much more than the absence of defects." Adding these dimensions of quality to the benchmark used by CISQ would bring the total tab for poor software quality even higher, potentially, by a substantial degree.

Cost of Poor Software Quality in the US, 2020 (Source: CISQ)



Implications for DevOps shops

Many major organizations today use a DevOps approach to software deployment that emphasizes close connections between developers and IT operations staff, managing the development of new capabilities through an Agile approach focused on rapid, small-batch releases that integrate business leads within the development process, or apply automation with CI/CD to accelerate and add structure to the development/deployment process. DevOps, Agile and CI/CD (as noted earlier in this

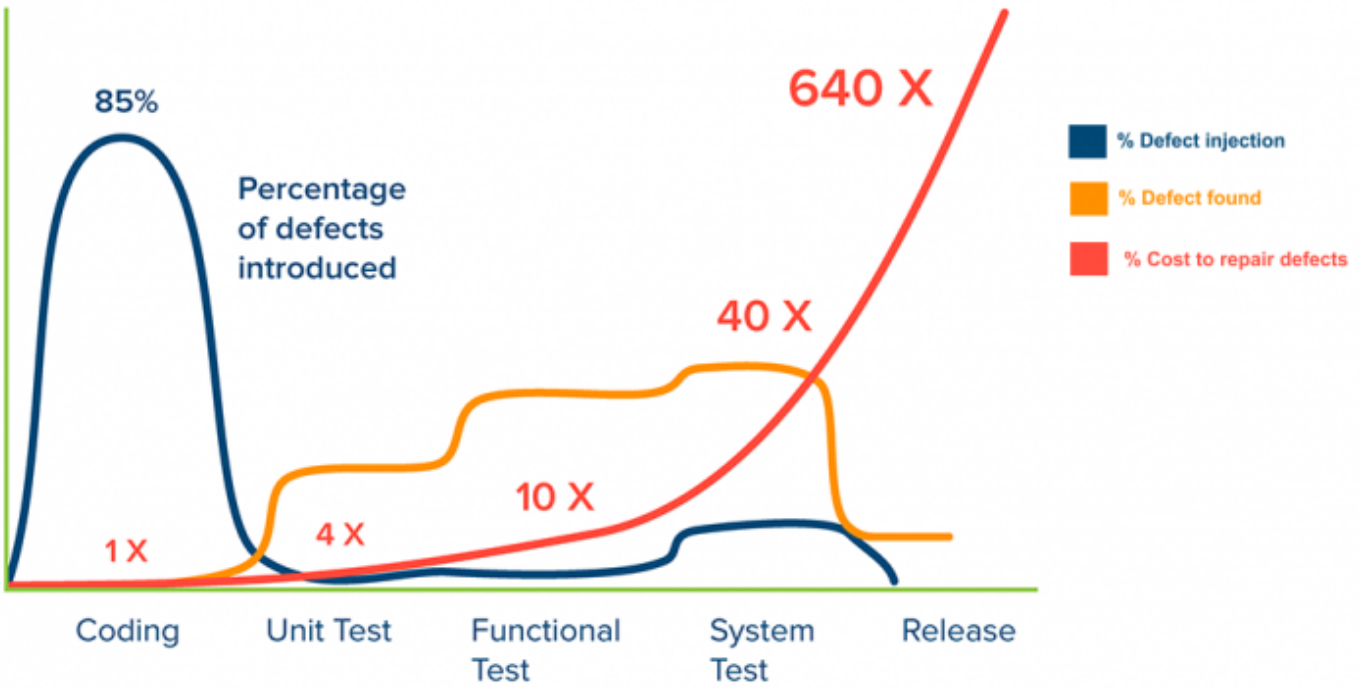
document) have independent meaning, but in practical terms, are often combined under a single heading. The collective impact of these approaches is substantial, delivering faster business value and increased customer satisfaction.

Much of the business benefit of the DevOps/Agile/CICD approach, though, is lost if organizational focus, metrics and incentives focus only on speed. Unless proper guardrails are established in advance, a “move fast and break things” approach to software development can become the source of costly quality issues. Two key areas of exposure are cybersecurity and user experience.

Cybersecurity and CI/CD

Consultants (and other services professionals) regularly try to manage customer expectations with the phrase, “you can have good, fast and cheap, pick two.” But from a cybersecurity perspective, the first objective may preclude both of the other options. Organizations that opt for “fast” rather than “good” may find expenses compounding over the life of a product. Research[5] has shown that while 85% of software defects are introduced in the coding stage, a majority are found later in the software development cycle – and at escalating cost; resolution of a defect after release can be 640 times the cost of resolution prior to initial unit testing.

Vulnerabilities injected, detected and repaired at different stages of the software development cycle (Source: Applied Software Measurement)



UX and MVP

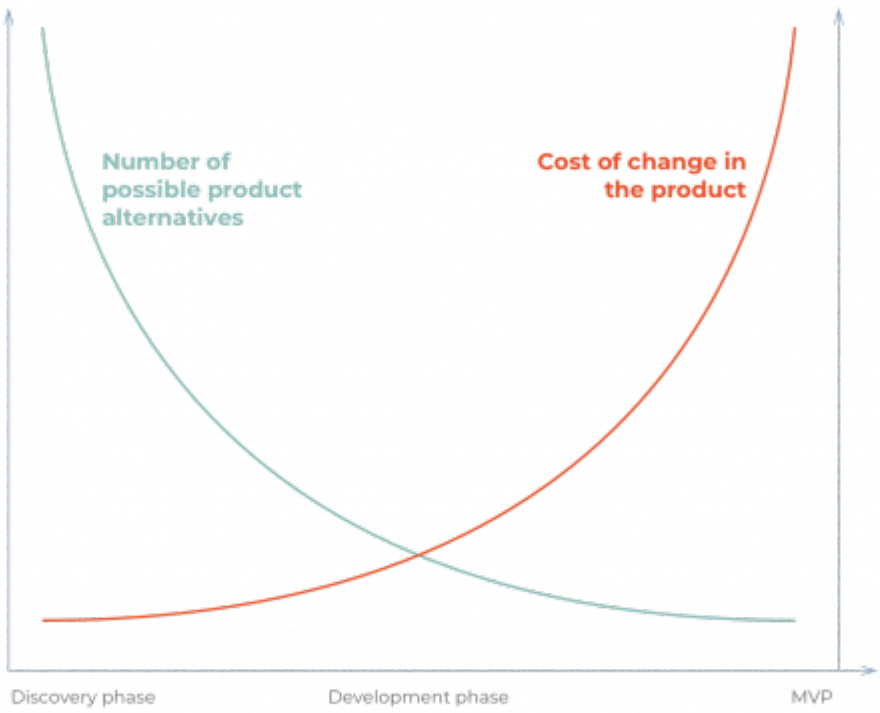
User experience is another key area that should not be overlooked in the rush for MVPs addressing specific business requirements. As with cybersecurity, the cost of addressing flaws in usability escalates tremendously through the course of the design cycle. The graphic below is an often-cited illustration of how design choices become more limited and cost of change increases through the design cycle – sourced here from a wattx.io blog post entitled "How investing in UX will save you money and time."[6] Text from the blog post states that "UX is often the first candidate when it comes to cutting time and costs on a project - if it is being considered for a project in the first place at all. As a matter of fact, most of our clients confess that their innovative ideas and projects are often executed without any involvement of user research and without ever talking to their target customers."

The post goes on to illustrate savings that accrue to firms that focus on users' experiences before embarking on the creation of an MVP – an approach that may be untenable for different reasons (e.g., competitive, political) within a specific environment. Perhaps more importantly, the post parallels the insight shown in the vulnerability data graphed above: "the cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to

100 times more than one identified in the maintenance phase.”

A focus on users’ experiences across the whole organization, especially when undertaking digital transformation efforts, will help you not only deliver it the right way, it will help you deliver the right thing.

Relationship between available design choices and the cost of design changes through the software development cycle (Source: wattx.io blog)



Recommendations

Given the prevalence of DevOps, Agile and CI/CD – and the business demands driving its use, and the business benefits delivered through its adoption – there is no question that adaptation will prove to be the best path forward for organizations in all sectors, and for the IT, UX and cybersecurity teams that support them.

The UX and cyber teams, though, need to find ways to ensure that the returns associated with early (in the cycle) action are captured by their operations. Stratascale recommends that organizations consider the following:

- Investigate tools for integrating security within DevOps and CI/CD: Mainstream cybersecurity suites do not typically extend to hardening the CI/CD pipeline. Cybersecurity teams are urged to explore tools that will automate critical capabilities at the front end of the CI/CD cycle.
- Don’t lose sight of the true goal: new/enhanced business value, not the release cadence for new software features. As Keith Instone, Stratascale’s Lead Research Analyst – Digital Experience, likes to remind us, “good digital experiences are one key reason *why* we are doing all this IT work, but just delivering new technology is not enough.” Pace is importance, but positive impact is the objective, not simply one of many metrics.
- Emphasize the need to couple Agile’s portfolio development benefits with the advantages of robust product rollout: Something to the effect of “Senior executives are accustomed to navigating situations which require a balance between breadth (speed of new product/capability introduction, range of new capabilities) and depth (quality). Optimizing the value of an investment portfolio is best achieved through a ‘breadth’ approach, while optimizing the returns obtained from each specific product depends in large degree on ‘depth’ objectives: reducing/addressing software defects as early in the lifecycle as possible and emphasizing capabilities that contribute to superior usability, security and performance.

In your operational context, FIUYMI may work as a shortcut to market presence, but it is a financially-dangerous fallacy when it comes to rolling out new products with the attributes (including, especially, cybersecurity and UX) that drive superior digital business returns.

Speed is critical to competitiveness, but obtaining optimal value from a software-dependent digital business infrastructure is also essential, due both to the magnitude of direct investment in development and the intrinsic reliance on software as the linchpin of digital business. MVPs play a central role in supporting operational agility, but as these MVPs evolve into core business infrastructure components, there is a need for a ‘speed to power’ transition that includes assurance that successful projects are hardened with the security and UX attributes that deliver optimal benefits.

Chris Hudson echoes guidance on integrating security within DevOps when he tells clients that “part of the MVP definition needs to include protecting the data (cybersecurity), as that needs to be a fundamental premise for all

products/features/software – from our thermostats (IoT) to the most sophisticated ERP systems…UX can and should evolve over time, but securing PII/PCI/HIPPA data is a core tenet" of public-facing software capabilities. Minimally usable and minimally valuable (as determined by users, not the executive sponsor) need to be part of the minimally viable formula.

In the end, core attributes like cybersecurity and user experience are, as the old saying holds, like sugar in a cupcake – if you leave it out while you are making the cupcakes, you can't compensate by doubling up (or in context, adding 640 times as much!) the quantity of sugar used in the frosting. DevOps needn't, and shouldn't, be a FIUYMI exercise: baking security and quality into the CI/CD process is crucial to aligning speed with benefit.

---

1. "Agile" and "DevOps" (and to some extent, "Continuous Integration/Continuous Deployment, or CI/CD) are distinct but closely-coupled concepts. In practice, the terms are often applied to the category of activities that span their collective use. Generally, they are viewed as an alternative to "Waterfall" methodologies, in which formal requirements are fully established as the first step in a highly-structured development initiative. In general, Agile/DevOps/CICD are viewed as delivering usable output much more quickly than is possible with the Waterfall methodology, and as being much more responsive to changing requirements or priorities.
2. Hening, Mirco. *DevOps for the Modern Enterprise.* IT Revolution, 2018.
3. Krasner, Herb. "The Cost of Poor Software Quality in the US: A 2020 Report." Consortium for Information & Software Quality, January 1, 2020.
4. Bhat, Manjunath, Hassan Ennaciri, Chris Saunderson and Daniel Betts. " Innovation Insight for Continuous Infrastructure Automation." Gartner Group, April 27, 2021.
5. Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality.* McGraw-Hill Education, 2008.
6. The concept shown in the graphic has been used for many years. The graphic itself dates back at least to 1994's *Cost-Justifying Usability* (published by Morgan Kaufmann; Bias, Randolph, and Deborah Mayhew, editors). The chart is based on data sourced from Roger Pressman's *Software Engineering: A Practitioner's Approach* (McGraw-Hill Education, 1992), which states "Assume that an error uncovered during design will cost 1.0 monetary unit to correct. Relative to this cost, the same error uncovered just before testing commences

will cost 6.5 units; during testing, 15 units; and after release, between 60 and 100 units.” This observation is in turn sourced to findings from an IBM research project conducted in 1981.

7. Bhat, Manjunath, Dale Gardner and Mark Horvath. “How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks.” Gartner Group, July 15, 2021.