ReThink:
A New IT Blueprint

Michael T. Nygard, Cognitect

THE STORY **IS THE SAME**

It doesn't matter what kind of project it is—ERP, CRM, MRP, web sites or personnel—the story is the same: $20 million over budget; 12 months late; budgeted $25 million for 18 months, spent $125M over 5 years.

With capital and time losses like that, it's not hard to see why we would organize IT to prevent them. Unfortunately, by trying to prevent the seen costs of project overruns, we've created unseen costs that can paralyze the entire company.

It comes as no surprise that information technology in most companies today is ghastly inefficient. This phenomenon has persisted over multiple decades, numerous management philosophies and many attempts to rein in costs and get projects under control.

Some companies are able to buck this trend and are radically successful. These are typically the startups, which can accomplish a tremendous amount with a small number of people in a short period of time. A few large organizations like Amazon and Netflix have been able to keep moving fast, but they are the exceptions.

I've looked at the problem of IT ineffectiveness from many different angles—outsourcing, PMO, enterprise architecture and more. All of these approaches have one thing in common: they aim to mitigate those huge project overruns by focusing on cost. Paradoxically, instead of helping make companies more competitive, they slow things down.

IT COMES AS NO SURPRISE
THAT INFORMATION TECHNOLOGY
IN MOST COMPANIES TODAY
IS GHASTLY INEFFICIENT
THIS PHENOMENON HAS PERSISTED
OVER MULTIPLE DECADES

01/ MANAGING ENTERPRISE RISK

According to the Standish Group's "Chaos Manifesto 2013," 60% of software projects fail. One particularly telling breakdown is the percentage of projects that succeed based on budget size. Nearly all $50,000 projects succeed. The success rate drops at $100,000, and dips again at the $1 million mark. When you get to the $5-10 million projects, the success rate is so near to zero that you can call it a rounding error.

Corporate infrastructure carries huge liabilities for the CIO and CFO—downtime costs dollars and careers—and compliance errors can (at least in theory) result in jail time. So it is natural to be cautious about new methods, and only allow the most trusted technologies to reach production. We can think of this as risk management by reducing the opportunities for error.

In fact, there are three ways to manage risk:

The first, reduce the number of opportunities for error, translates to fewer application deployments, fewer changes and slower delivery.

The second way to manage risk is to reduce the probability of an error occurring at each of those opportunities. This is the aim of every review process and every stage gate process.

The third way to manage risk, and the one that I believe is the most effective, is to reduce the cost of an error, should one occur. This is accomplished in two ways. The first is to break an application into small failure domains, which allows a portion of an application to fail without impacting other parts of it. The second is to ensure that errors are caught and corrected as quickly as possible.

Reducing the cost of errors, as opposed to reducing the number of errors is key. If we want the competitive benefits offered by agile development, continuous delivery, and DevOps, then we must expect to increase drastically the number of opportunities for error. Thus, we must make corresponding reductions in the cost and probability in order to get the promised gains.

The first approach to risk management, to reduce the number of opportunities for error, still dominates the thinking in today's modern enterprise. We've had 15 to 20 years of trusted methods: ITIL, CoBIT, enterprise architecture, and the PMO. And yet projects continue to stagnate. We pull CIOs in multiple different directions. The drumbeat continues: IT needs to align with the business. And the response? "What the hell do you think I'm trying to do?"

02/ IT'S ABOUT TIME

We must refocus our attention from control to speed: speed of delivery, error detection and recovery. Outsourcing, insourcing, PMO—our accepted IT constructs—all add project delays. For example, one of the dominant factors in a project timeline is the speed of decision-making. Because decision-making responsibility is diffused across so many people in an organization, it sometimes happens that no one can make a decision and progress comes to a screeching halt.

Delays also arise when teams are imbalanced. I have witnessed a meeting where there were eight project managers (from different reporting chains) and two developers. They brought in an agile coach to understand why the project wasn't moving very fast!

The biggest offender is the queue. We tend to worry about keeping people busy all the time, when we should worry about keeping the work busy. Any time the work sits idle while people or machines are busy, it is in a queue. Want to know how long it will really take to finish a project? Don't add up the task durations, add up the queues.

Queues arise when a company tries to control expenses by keeping people busy, overburdening and "time slicing" them. For example, if someone is busy

The Paradox of Busy and Effective

When a person is maximally busy, you're virtually guaranteeing that they will be ineffective. Outsourcing as labor arbitrage, which focuses on ensuring that people are busy 100 percent of the time, is a good example. When you look at the work being performed, these people may be churning away every hour, but the piece of work you need may sit for a long time before it gets to the head of their queue. This approach is backwards: we need to track utilization on the work rather than the worker.

Manufacturing environments have observed this, and focused on activities such as value stream mapping to identify wasted time where work sits idle. To reduce waste, you manage utilization downward to the point where you can complete work in a short time. For example, if new work arrives at a certain rate then you may want a 70% utilization from that person so that they are available when the work arrives. Outsourcing drives you in exactly the opposite direction.

Matrix management is another technique that creates delays. If you only need half a database administrator on a project, the assumption is that the person will be available when you need them. However, if the schedule is not a fixed set of hours, odds are they won't be because they're busy working on something else.

These high utilization models produce long queuing time, which in turn produces high utilization. The effect on projects is that the timeline stretches out by factors and the costs multiply.

with five projects on their desk, all the projects move slower. The odds are low that this individual will be available to work on your project when you want, as four projects sit idle at any given moment. Further, when a company reduces the number of people working on projects to save money, it slices everyone's time even thinner. This always adds up to delays, which drive the cost of everything up—the exact opposite of the intended effect.

Some queues are easy to see. If your development team has to wait for the next architecture review board meeting, then they are waiting in a queue. Other queues are subtler. Have you ever held a meeting where a critical decision maker wasn't there, so you rescheduled? Queuing. Had to schedule a meeting for later in the week because there aren't enough conference rooms? Queuing.

One of the biggest, yet least visible queues is the one between "the business" and "IT." At that boundary, business needs first get bundled together into projects. It's common to find multiple years' worth of work waiting to cross that semi-permeable membrane.

The effect of delays across the organization doesn't just add up, it multiplies. When one project takes 20% longer and another also takes 20% longer because they're both competing for the same resources, it creates bottlenecks that result in both projects actually taking about 50% longer.

It's a cruel paradox: cost saving and risk-reduction methods introduce oversight, and oversight always adds centralization, review and delay. The bottom line: when you try to cut costs, you go slower and slow is expensive. When you focus on going faster, it is actually cheaper because it is difficult to go fast and spend a lot of money.

03/ ISOLATED SOLUTIONS: LIMITED BENEFITS

The manufacturing industry recognized queues as a disease decades ago. They adopted practices such as lean manufacturing to identify waste and reduce delays, with great success. We can view agile software development and DevOps as two ways to apply the same concepts to IT.

What I find is that many large companies following these methods don't truly understand why, so they only implement part of the process, and thus don't reap the full benefits. An example of this is to implement agile software development when you're on an annual budget cycle. When a project must go through months of approvals before you can get started, the IT organizational structure itself limits this high-velocity method from realizing its full benefits. Another example is trying to get things done quickly, but with 100% staff utilization, you experience long delays due to queuing.

THE BIGGEST OFFENDER IS THE QUEUE. WE TEND TO WORRY ABOUT **KEEPING** PEOPLE BUSY ALL THE TIME, **WHEN WE SHOULD WORRY ABOUT** KEEPING THE WORK BUSY.

Agile software development, DevOps, the cloud and lean thinking all offer advantages.

However, today they are typically being applied within the constraints of the old IT blueprint: isolated versus interrelated solutions built on a monolithic infrastructure where the cost of errors is high, all embedded in a matrix of centralized oversight.

If we continue down the path where IT manages everything from phones to web apps, people are matrix managed, we use median technologies that drive all performance to the middle and build centralized infrastructures that demand high oversight, we can continue to expect project delays, mounting costs, marginal results and missed opportunities.

04/ THE NEW IT BLUEPRINT

It's clear that we need to take a fresh look at the role of information technology in the modern corporation. We're seeing two things that make me believe the time for change is here. First, even the most conservative of companies is adopting languages and frameworks that offer better development speed. Second, many companies are already experimenting with different architectures and organizational models.

We need a new technology blueprint that empowers the CIO to reinvent the IT department as we've known it: reallocate responsibilities, allow the entire organization to move faster, make better use of company assets and compete more effectively.

IT should be responsible for running a platform as a service. It should not be responsible for applications—the users who are closest to and most effected by them should write and control them. You don't need a business analysis project when you're sitting next to the people who know what they need. Today, we call the applications that come out of this scenario "rogue." Moreover, we rein in the person who develops them before they do harm.

The opposite needs to happen. We need to further enable these people within an architecture that makes it safe for the rest of the company. Some companies are looking to microservices as an alternative to achieve this. Applications with a microservice architecture consist of a set of narrowly focused, independently deployable services.

I don't believe that "micro" is the key. Even "services" is not the key. Most teams dabbling in microservices are going to fail because they focus on what the service does or knows. Instead they should focus on what the service's consumers do when the service breaks. Such a resilient architecture, broken down into isolated failure domains, reduces the need for oversight and it's attendant queues.

In a resilient architecture, part of the application can fail but the rest of it continues to function. The architecture you employ to do this looks crazy to the enterprise architect because it relies on things like data duplication, which they strive to eliminate. In fact, this type of architecture looks quite alien to most people, but once you have it in place, each individual group can move at its own speed, deploy on its own, and indeed fail on its own—without having a negative impact on anything else. The infrastructure and architecture that enables this is all possible. This is managing risk by reducing the cost of errors.

Two Approaches to Preventing Downtime

Everyone knows you need to reduce downtime—it's expensive. For example, a single hour of downtime for a company the likes of a BestBuy or Walmart costs $20 million in revenue.

One way retailers try to avoid downtime is by doing all the preparation and making software changes throughout the year. Then, they make no changes during the holiday peak-sales season, and have a large operations team in place to mind the store.

This approach can work for a while, but once an organization reaches a certain size, downtime is inevitable. (Every tech publisher has a "Black Friday" template for these outage stories. It's the same story every year—they simply cut and paste to change the name of the company.)

The other approach is to make changes all the time, and write the software so it continues to run as we make these changes. Making changes in this manner is akin to partial failure. If you can survive partial failure, you are much more likely to avoid downtime during your peak season.

The architecture and organizational changes are inseparable. This is the rock that many agile adoption projects have foundered on: the organizational structure, the architecture, and the processes mutually reinforce and constrain each other.

The result of the new blueprint is an anti-fragile enterprise.

We're not simply cheerleading new technologies here. Rather, we're taking a practical approach, applying them in ways that bring together the things that work from the best of them, while we still allow for compliance and mitigate risk. There are companies doing this successfully today. These are the companies that are going to survive and succeed. Those who don't get onboard will perish.

MICHEAL T. NYGARD
vice-president cognitect

You can find Michael coding, writing, speaking, or thinking about how the Universe works. He has lived with systems in production and learned hard lessons about the importance of operations. Highly-available, highly-scalable commerce systems are his forte.

ABOUT THE AUTHOR

Throughout his career, Michael has strived to raise the bar and ease the pain for developers across the country. He has been a professional programmer and architect for nearly 20 years. During that time, he has delivered running systems to the U. S. Government, the military, banking, finance, agriculture, and retail industries. More often than not, Michael has lived with the systems he built. This experience with the real world of operations changed his views about software architecture and development forever.

Michael has written and co-authored several books, including "Release It!," "Beautiful Architecture," "97 Things Every Software Architect Should Know" and "Java Developer's Reference." He is currently working on his next book.

ARCHITECTURE AND ORGANIZATIONAL CHANGES **ARE INSEPARABLE.** **THIS IS THE ROCK THAT** MANY AGILE ADOPTION PROJECTS **HAVE FOUNDERED ON.**

cognitect

phone: 919.283.2748 | email: info@cognitect.com | www.cognitect.com | @cognitect

AGILE
ITERATIVE SOFTWARE DEVELOPMENT.
EXPERTISE.

Experience the Difference.