2013
Issue 2

06
The evolution from lean and agile to antifragile

26
Making DevOps and continuous delivery a reality

46
A CIO’s DevOps approach to resolving the agility-stability paradox

DevOps: Solving the engineering productivity challenge

John Esser
Director of Engineering Productivity and Agile Development, Ancestry.com
Contents

2013 Issue 2

Features

06

The evolution from lean and agile to antifragile
New cloud development styles have implications for business operations, too.

26

Making DevOps and continuous delivery a reality
Automation, thoughtful choices, and a well-designed workflow are pivotal to making good progress.

A CIO’s DevOps approach to resolving the agility-stability paradox
Real process change requires a break with the past.

46
18
Reconsidering risk in a continuous delivery environment
Continuous Delivery author Jez Humble says culture, rather than risk, is the real obstacle to transforming IT.

22
How established enterprises are moving toward DevOps
Vijay Gopal of Nationwide Insurance discusses DevOps within the context of an insurance company’s view of risk.

38
Using DevOps and continuous delivery to move at the pace of the business
John Esser of Ancestry.com discusses his role in the company’s transformation efforts.

42
Blueprinting continuous delivery inside the enterprise
Lothar Schubert and Laurence Sweeney of CollabNet compare different enterprise maturities and ways to bring each type closer to a continuous delivery model.

58
An aerospace industry CIO’s move toward DevOps and a test-driven development environment
Ken Venner of SpaceX explores agile development’s latest evolution.
Acknowledgments

Advisory
Technology Consulting Leader
Tom DeGarmo

Chief Technologist
Chris Curran

US Thought Leadership
Partner-in-Charge
Rob Gittings

Strategic Marketing
Natalie Kontra
Jordana Marx

Center for Technology and Innovation
Managing Editor
Bo Parker

Editors
Vinod Baya
Alan Morrison

Contributors
Phil Berman
Galen Gruman
Todd Johnston
Harshul Joshi
Keith Katz
Bud Mathaisel
Leslie Moore
Bill Roberts
Robert L. Scheier

Editorial Advisor
Larry Marion

Copy Editor
Lea Anne Bantsari

Transcriber
Dawn Regan
Industry perspectives
During the preparation of this publication, we benefited greatly from interviews and conversations with the following executives:

Chad DePue
Founder
Inaka

Brian Doll
Vice President, Marketing
GitHub

John Esser
Director, Engineering Productivity and Agile Development
Ancestry.com

Carlos Escapa
Senior Vice President and General Manager, Disaster Recovery Solutions
PHD Virtual

Toby Ford
AVP, Cloud Technology, Strategy and Planning
AT&T

Vijay Gopal
Vice President and Enterprise Chief Architect
Nationwide Insurance

Ed Haletky
Managing Director
The Virtualization Practice

Jez Humble
Principal
ThoughtWorks Studios

Andy Kelk
CIO
iProperty Group

Cliff Moon
Founder and CTO
Boundary

Craig Muth
Senior Engineer
Homerun.com

Pat O’Day
CTO
Bluelock

Charles Oppenheimer
Founder
Prizzm

Ranga Rangachari
Vice President and General Manager, Storage
RedHat

Marko Rodriguez
Graph Systems Architect
Aurelius

Lothar Schubert
Senior Director, Solution Marketing
CollabNet

James Staten
Vice President and Principal Analyst
Forrester

Laurence Sweeney
Vice President, Enterprise Transformation
CollabNet

Bryan Taylor
Principal Architect
Rackspace Managed Hosting

Andy Thurai
Chief Architect and Group CTO – ASIP
Intel

Matt Tucker
CTO and Co-founder
Jive Software

Ken Venner
CIO
SpaceX
establish standard ways of working, or operational stability, designed not to change at all. When change is thrust upon them, companies create long-running projects that bundle together a number of changes, introduce those changes without embedding real-time feedback loops to measure and respond to outcomes, and seek to establish a new stability in operations until a new bundle of changes is implemented at some future time.

The common mental model for the design and operation of the modern organization over time is defined by long periods of stable operations, then major, disruptive, managed-from-the-top changes to operations with insufficient synchrony between action and outcome to accommodate control, followed by another long period of stable operations.

This model permeates most everything organizations do for “change management.” Consider the best practices established by many IT organizations that follow the Information Technology Infrastructure Library (ITIL) bible:

“Goal: Ensure that standardized methods and procedures are used...
for efficient and prompt handling of all changes, in order to minimize the impact of Change-related incidents upon service quality, and consequently improve the day-to-day operations of the organization.”

So far, so good. Who could complain about “efficient and prompt” and “improve the day-to-day operations?” But most companies turn this positive vision into a slow-moving, nonresponsive situation where permission to change anything must be run past a Change Advisory Board. On the surface, the goal seems admirable: avoid pain from change that disrupts operations. And if change is deployed very sporadically in huge bundles with few opportunities to accurately assess—in real time—the impact of that bundle of change, then a bureaucratic, top-down process is probably appropriate.

This issue of the Technology Forecast asks the question: “Is there some other way?” Going in, PwC assumed the answer had to be “yes,” because web-scale companies are accelerating through disruptive periods of growth with some pain (that they anticipate, embrace, and learn from), and there are few examples of catastrophic failure. What enables such success with change management when change is a constant? What can other companies learn from their approaches?

Business processes must be adaptive enough to deliver solutions for today’s problems and opportunities, not yesterday’s. And large enterprises today are often full of good ideas that would result in better processes—but they confront legacy IT systems and project management procedures that take too long to deliver the infrastructure and application changes needed to support process changes. Major culprits include the way IT manages change and the risks that system changes create for stable operations. IT reaches for stable operations by minimizing the frequency of code changes.

Contrast that approach with how web-based companies operate, even very large ones such as Facebook, Google, and Netflix. These companies strive for continuity of operations by embracing change. Some introduce new code to their production systems many times a day. The counterintuitive outcomes for web-scale companies are systems that are much better aligned to what the business needs today and more responsive to what the business will need tomorrow.

Large enterprises are beginning to learn from what’s happening in these web-scale companies and are trying similar approaches internally. They’re discovering that an “antifragile” definition of stability is emerging that builds on existing agile development principles. The notion of highly adaptive “antifragility” applies not only to IT specifically, but the business as a whole.

The article, “The evolution from lean and agile to antifragile,” on page 06 considers the many industries experiencing an unprecedented amount of stress and disorder. How can enterprises turn stress and disorder into a positive? One answer may lie in how some web-scale companies make frequent daily changes a routine part of their cloud-based services. That pace of change becomes possible at a business design level using an advanced form of agile development, and that ability to quickly shift direction can move an agile company into a hypercompetitive, more antifragile state. These companies are in some ways antifragile because they invite and can immediately adapt to shocks.

“Making DevOps and continuous delivery a reality” on page 26 explores the practical application of continuous delivery. In continuous delivery, frequent—even daily or hourly—incremental code releases replace what previously transpired across weeks or months. Enterprises in established industries are now moving toward more frequent code releases with many of the same tools and techniques in use at web-scale companies. The fact is that startups and established organizations alike are using new tool chains to accelerate code delivery and deployment while paradoxically reducing the unintended introduction of bugs and the need for code fixes.

“A CIO’s DevOps approach to resolving the agility-stability paradox” on page 46 sheds new light on what stability is in the era of the cloud and how it is best approached. For decades, CIOs have applied methods and adopted tools that lead to at least some form of stability. But CIOs must change the IT mindset about how to achieve stability. The key is to adopt a DevOps philosophy that emphasizes speed and efficiency without sacrificing true stability.

This issue of the Technology Forecast also includes interviews with executives and subject matter experts who have been at the forefront of development in this area:
- Jez Humble of ThoughtWorks Studios
- Vijay Gopal of Nationwide Insurance
- John Esser of Ancestry.com
- Lothar Schubert and Laurence Sweeney of CollabNet
- Ken Venner of SpaceX

Please visit pwc.com/techforecast to find these articles and other issues of the Technology Forecast online. If you would like to receive future issues of this publication as a PDF attachment, you can sign up at pwc.com/techforecast/subscribe.

As always, we welcome your feedback and your ideas for future research and analysis topics to cover.
Technicians digitize old books at Ancestry.com and store the content in a searchable database.
The evolution from lean and agile to antifragile

New cloud development styles have implications for business operations, too.

By Alan Morrison and Bo Parker

Mike Krieger and Kevin Systrom, two entrepreneurial software developers with modest prior knowledge of back-end systems, built and launched Instagram in 2010 on a single server from one of their homes. Then they stood back to see what would happen.

On the first day, 25,000 users signed up for the mobile photo-sharing service. Within two years, Instagram had 2 million users. It acquired 1 million users in just 12 hours after it began offering an Android version. By May 2013, nine months after Facebook acquired Instagram for $1 billion, the photo-sharing service had more than 100 million active users per month and continued to show substantial new growth quarter to quarter. (See Figure 1.)

As user sign-ups skyrocketed after the launch, the founders quickly decided to move the service to the Amazon cloud. The cloud, as will be described later, offered the scalability needed, but Instagram’s rapid growth also demanded they revisit how the entire system operated. They recognized that trying to achieve the absolute stability of a traditional enterprise system was unrealistic given the exponential growth. Instead, they designed the system to take advantage of the uncertainty created by the pace of growth even as the growth itself forced changes to core elements. With this approach, they solved a succession of scaling problems one at a time, as they arose, using open source databases and various data handling and other techniques identified with the help of open developer communities.

In April 2012, Krieger acknowledged that they had learned the real meaning of scaling: “Replacing all components of a car while driving it at 100 mph.”

Like the Instagram founders, several other web entrepreneurs have taken advantage of cloud development, infrastructure, and operations capabilities that many established enterprises don’t use and only vaguely understand. These capabilities are consistent with an emerging systems management concept called antifragility, which has significant business implications for all kinds of enterprises, large or small, new or well established.

This issue of the Technology Forecast describes the emerging IT development, infrastructure, and operations methodologies in the broader context of strategic and operational enterprise change initiatives. PwC forecasts that agile methods in businesses will force modifications in IT, leading to the adoption of change management techniques in IT similar to the antifragile approach that web-scale companies use to manage rapid and continuous change.

This first article discusses the antifragile approach and how it embraces continuous, disruptive change. It positions antifragility in the context of agile and lean methodologies, detailing how the principles underlying these methods remain in the antifragile approach. The article also positions antifragility as a catalyst for broader organizational change, and it traces the link between the pace of change that becomes possible using agile methods at a business design level and the potential disconnect if IT cannot keep pace.

The second article covers tools and techniques that broadly support the continuous delivery and deployment required of antifragile systems. Facebook, LinkedIn, Google, Yahoo, and others have needed to develop tools that support this approach. The good news is they generally have made these tools available to others as open source technology, and a network of support vendors is beginning to help companies adopt them.

The third article describes the CIO’s challenges in taking on the demand for a more agile business, including the cultural changes needed within IT. One key question is how do you get comfortable with the seeming paradox of simultaneous constant change and stability in production systems? The answer lies in the adoption of DevOps methods.

**Antifragility: What it is**
The Instagram example might seem like just another typical Silicon Valley startup success story, but its broader significance is the level of web-scale responsiveness that a handful of people were able to quickly discover and demonstrate. For Instagram, scaling meant anticipating, observing, and fixing performance problems when they arose as quickly as possible. They ended

---

**Sidebar**

**Testing as a service**

Multi-application test strategies offer the advantage of testing at a layer of abstraction above the application software development life cycle level. These strategies can be used even if several software development life cycle systems are in use.

With typical managed service testing, companies pay for resources whether they use them or not. Cloud-based testing models provide companies greater flexibility to pay for resources, frameworks, and tools as they are needed. Enterprises that have a multi-app platform strategy can use a testing-as-a-service (TaaS) provider on a per-use basis. When the TaaS provider executes against the multi-app platform strategy from the cloud, resources can be engaged from any location, and testing artifacts and test data can be shared. In addition, the TaaS approach contains environment costs, because test environment provisioning and de-provisioning occur via the cloud.

**Figure 1: Rapid social network growth at Instagram**

<table>
<thead>
<tr>
<th>Adoption rate</th>
<th>Facebook 100%</th>
<th>Twitter 90%</th>
<th>Google+ 80%</th>
<th>Pinterest 70%</th>
<th>Instagram 60%</th>
</tr>
</thead>
</table>

Source: Simply Measured, May 2013

---

In April 2012, Mike Krieger of Instagram acknowledged that they had learned the real meaning of scaling: “Replacing all components of a car while driving it at 100 mph.”
up developing a giant feedback-response loop; rapid responsiveness made it viable. This level of responsiveness has only recently become possible due to cloud computing. Negotiating the learning curve quickly to achieve a level of consistent responsiveness is fundamental to antifragility, and it is a capability sorely needed in today’s enterprises, whether by cloud developer teams or business more generally.

The term “antifragility,” coined by Nassim Nicholas Taleb in his book Antifragile: Things That Gain from Disorder, is defined by the source of its strength. Like the human body, whose immune system gets stronger from exposure to germs and diseases, the antifragile system improves or responds positively when shocked.

While fragile systems are easily injured and suffer from volatility, antifragile systems grow stronger in response to volatility. So-called robust systems remain unchanged. (See Figure 2.)

The concept of antifragility becomes clearer when compared and contrasted with stability. As Taleb describes it, antifragile is beyond stable, beyond robust; stable and robust systems resist shocks but stay the same. In contrast, Taleb describes antifragile systems as those capable of absorbing shocks and being changed by them in positive ways. The key insight is that stable systems, because they don’t change, eventually experience shocks large enough to cause catastrophic failure. Antifragile systems break a little all the time but evolve as a result, becoming less prone to catastrophic failure.

---

Taleb, a former hedge fund manager and derivatives trader, argues that antifragile systems gain strength from volatility. They are the exact opposite of fragile systems. He spent years pondering the vulnerability to volatility and, in particular, infrequent catastrophes sometimes called “black swan” events. In *The Black Swan*, his earlier best-selling book, Taleb noted that many banks and trading companies used stable models that worked most of the time but were unable to predict or account for the rare event of house prices falling throughout the United States, leading to the recent financial crisis. When the magnitude of change stays within a normal range, robustness can be a state that seems resilient. During periods of unusual change, only the antifragile organizations prove to be resilient.

The concept of antifragility evolved from the socioeconomic context with which Taleb was most familiar, not from cloud computing. But it’s in the cloud that the concept may have its broadest practical application to date.

Rob England, author of *Basic Service Management*, has made the connection between the concept of antifragility and the design and operation of today’s IT systems. “If an organisation’s IT is unstable, we can move from a fragile condition to a spectrum of better states, ranging between antifragile and robust,” he writes. Fragile organizations seek to become less fragile or—ideally—antifragile. Organizations that become robust may still be fragile.

Life, Taleb says, is not as predictable or explainable as the rationalists would have us believe; instead, simple sets of rules help us to navigate through a complex and constantly changing landscape. He argues, “We have been fragilizing our economy, our health, education, almost everything—by suppressing randomness and volatility. If about everything top down fragilizes and blocks antifragility and growth, everything bottom up thrives under the right amount of stress and disorder.”

Business uncertainty will demand agile, antifragile methods

The idea that businesses might be experiencing an unprecedented amount of “stress and disorder” should come as no surprise. CEOs and other senior executives consistently describe uncertain future business conditions as a key concern. Some of the biggest drivers of uncertainty include:

- **Narrower windows of opportunity:** Companies offering products and services with distinctive value propositions are experiencing shorter windows of pricing power before competitors commoditize the market.

- **More convergence:** Companies are expanding their addressable markets by moving into adjacent areas, often becoming intense competitors with enterprises they formerly bought from or sold to, forcing incumbents to bundle capabilities and expand their value footprints.

- **Slower growth in traditional markets:** Developed economies have been in a deep slump for several years with low expectations for future growth. Developing economies offer the greatest potential for new revenue, but multinationals often cannot meet the lower price points or access distinctive channels to market in these economies.

Like biological organisms, antifragile systems adapt and evolve in response to stress and changes to their environment.

---


Greater need for business model change and additional information services: These uncertainties are leading many companies to consider radical changes to their business models, such as bundling services with products and offering total solutions designed to help customers achieve their goals. (See Technology Forecast 2013, Issue 1.) Other companies are unbundling their vertically integrated offerings and introducing application programming interfaces (APIs) that encourage third parties to build new offerings based in part on these unbundled capabilities. (See Technology Forecast 2012, Issue 2.) In each case, the people, processes, technologies, leadership, organizational designs, and strategies in these enterprises are experiencing tremendous stress. That’s because large companies have scaled up by developing highly capable but siloed functional domains that minimize cross-organization information and interactions for efficiency purposes. This approach is ideal in a stable world, but in today’s dynamic markets these flows can’t be redesigned and redeployed fast enough. Most companies respond by using traditional business process reengineering methods that explicitly or implicitly seek a new, stable operating environment. In many cases, by the time that new stable design gets deployed, the market has moved on.

Management is becoming more and more aware that traditional approaches to business redesign can’t work in rapidly changing business environments. They recognize that employees who have the ability to work across their functional domains should be able to effectively respond. Often, existing practices for discovering and designing optimum solutions perpetuate the silos.

Toward a fully digital business

More and more, the quickest way to a broader customer base is through new digital touch points. Mobile is the fastest growing part of this shift, but it’s not the only part. Mobile payment transaction values will grow 44 percent worldwide in 2013, according to Gartner, rising to $236 billion from just under $201 billion in 2012. E-commerce as a whole will reach $1.22 trillion in 2013, according to eMarketer, up more than 17 percent from $1.047 trillion in 2012.

How do enterprises take better advantage of the rapidly growing online environment? By going to where the customer is and by being flexible enough to lead in an environment that constantly changes. Competitive advantage hinges on the ability to be responsive to customers, and that means providing user-centric interfaces, rich information services via application programming interfaces (APIs), consistently compelling value propositions, and a simple way to transact business on whatever kind of device a customer has.

Cloud development methods and the DevOps approach to frequent delivery and iteration are at the core of the digital customer touch-point challenge, because they’re a microcosm of the way a fully digital business operates. Companies that study how talented DevOps teams work can use the insights to inform the business side of their customer engagement models and their operating models generally. A fully digital business runs on what PwC calls the New IT Platform—a way to enable the business to immediately shift direction and be as responsive as new online customers demand.

The pivotal capabilities that enterprises need to tap into—whether sourcing expertise, co-creating with customers on a new product line, or motivating a purchase by a new customer with the help of gamification—are feasible at scale only when the business has made a full digital transformation. (See “Capitalizing on the strategic value of technology” at http://www.pwc.com/us/en/increasing-it-effectiveness/index.jhtml for more information on PwC’s New IT Platform.)


Company size and history can limit the ability to adopt DevOps

DevOps is a working style designed to encourage closer collaboration between developers and operations people: Dev+Ops=DevOps. (See the article, “Making DevOps and continuous delivery a reality,” on page 26 for more detailed information on DevOps methods.) Historically those groups have been working more or less at cross-purposes. DevOps collaboration seeks to reduce the friction between the two groups by addressing the root causes of the friction, making it possible to increase the volume and flow of production code, to improve the efficiency of developer teams, and to reduce the alienation of the ops people who must guard the stability of the system.

One of those root causes of friction is poor workflow design. DevOps encourages extensive automation and workflow redesign so developers can release small bits of code frequently (in a more or less continuous delivery cycle) and yet not disrupt the operational environment in doing so. The workflow includes buffers, compartmentalization, and extensive monitoring and testing—a very extensive and well-designed pipeline, but also a rich feedback loop. It’s a very test-driven environment. When the small bits of code get deployed, the individual changes to the user experience tend to be minor or limited to a small subaudience initially.

The use of a test-driven, frequently evolving DevOps approach lays the groundwork for antifragility.

However, many companies don’t have the ability to adopt DevOps quickly, reap the benefits, and move toward an antifragile state. Compare the way enterprises adopt a DevOps approach with what an Instagram did (a cloud native, as in Figure 3) from the start or an Ancestry.com (a pre-cloud enterprise) did within a few years. Instagram is a couple of years old, and Ancestry is a couple of decades old.

By contrast, Nationwide Insurance has been around since 1926 and has presumably had computer systems for more than 50 years. In spite of that legacy, the company has adopted some agile development practices and is looking toward DevOps.

<table>
<thead>
<tr>
<th>Legacy mainstream enterprises</th>
<th>Pre-cloud enterprises</th>
<th>Cloud natives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have many considerations and business priorities that must be addressed when implementing DevOps.</td>
<td>Have a few legacy systems to consider, and with appropriate effort can achieve significant transformation to DevOps.</td>
<td>Can start with DevOps and cloud from the beginning and establish the antifragile environment in its purest form.</td>
</tr>
</tbody>
</table>
How antifragility fits in with other change management theories

Enterprise process optimization has a heritage that goes back at least to Gustavus Swift and his meat processing efficiency breakthroughs, which inspired Henry Ford’s automobile production assembly lines. In the postwar era, the key processing efficiency breakthrough was statistical process control, which led to Kaizen (a continuous improvement philosophy) in the 1980s in Japan, waterfall development, and the rise of IT service management in the 1990s. Since that time, IT development has adopted a version of the Lean Enterprise and sought effectiveness and efficiency through varieties of agile development. (See Figure A.)

Figure A: Antifragility as an outgrowth of agile and previous change management theory

Vijay Gopal, vice president and enterprise chief architect at Nationwide Insurance, acknowledges that the company has “some antiquated systems that have not been developed with the degree of consistency or using the principles that we would like to have. We’re investing significantly to move toward that goal. But we also need to keep in mind that we operate in a regulated industry, and we can’t release code that could compromise customer information or some of the financial and sensitive data that we have. We would need to prove this agile development in some of the noncritical areas of our business first.”

An older and larger enterprise must deal with many, many issues when trying to make changes.
An operating environment designed for test drives
Plan to write the test before you code and allocate more time to testing in general.

An ability to scale change efforts in parallel
Venner points out that cloud infrastructure makes it possible to manage several ongoing development streams simultaneously. “Virtualization lets you spin up a large number of instances of your application environment. Therefore, the ability to have multiple parallel scrums going at any one time is relatively cost-effective and relatively simple to do nowadays.”

Antifragility starts with responsiveness and the ability to make changes at will; cloud both enables and requires that responsiveness. The iterative, highly responsive, and collaborative development model used by companies such as Instagram is inspiring other users of the public cloud. “My development team is aware of how the build process takes place and how test-driven development is evolving at companies like Facebook, LinkedIn, and Google,” Venner says. “Not all of it is applicable when dealing with spaceflight, but our goal is to take best practices from multiple industries to create the best possible solutions for SpaceX. Right now we release at least once a week, sometimes midweek, and we’re trying to move toward a continuous build and integration environment.”

Web-scale companies are not especially protective of their newest development, infrastructure, and operations philosophies or even their code.

Moving toward antifragility: SpaceX
Some enterprises in established industries are taking steps toward an antifragile version of stability, even though they don’t use that term yet. Space Exploration Technologies (SpaceX), a private aerospace company founded in 2002, focuses on making commercial space travel viable by reducing launch vehicle costs. The following paragraphs describe some of the notable characteristics of the SpaceX approach.

A business strategy built on intentional disruption
SpaceX, led by PayPal and Tesla co-founder Elon Musk, expects and indeed tries to create market disruption. CIO Ken Venner says companies should expect and prepare for rapid industry change: “The guys who are successful today may not be paranoid enough to figure out that the world can get changed on them. There is a mechanism by which the world will get changed, and features will come out at a pace they can’t keep up with. That will be a problem.”

A rapid change management approach focused on small iterations, scrums, and sprints
SpaceX uses modest-sized teams or “scrums” of business analysts and developers who “sprint to the finish line” to deliver small collections of new features that can be added to the system in an automated and straightforward manner. They then repeat the process multiple times, responding to feedback that generates new insights.

Frequent IT collaboration with business units
“You’re interacting with the users more often and you’re getting stuff set up faster,” Venner notes. “So you’re all starting to quickly see, ‘If I make this change over here, what potential impact could it have somewhere else in the organization?’ You’re showing users things versus talking it through, and they’re quicker to figure out that it may or may not work.”

Encouraging antifragility: Chaos Monkey
Web-scale companies are not especially protective of their newest development, infrastructure, and operations philosophies or even their code. Netflix, LinkedIn, Facebook, Google, Twitter, and many others all share code and openly discuss how to update their sites and improve site performance.
For example, Netflix shared Chaos Monkey with the open source community. Netflix designed Chaos Monkey to shut down cloud services at random to test the fault tolerance and redundancy of the services’ customer-facing systems. The description (not to mention the name) of Chaos Monkey makes it seem like a catastrophically disruptive tool, but in a cloud-computing context, the tool is valuable. Like biological organisms, antifragile systems adapt and evolve in response to stress and changes to their environment. Chaos Monkey applies a helpful kind of stress in the cloud.

In 2010, Netflix executives pointed out that their adoption of Amazon Web Services (AWS) forced them to think differently about customer-facing services. The core of that new awareness was to eliminate dependencies on services delivered with varying degrees of latency. As John Ciancutti, formerly of Netflix and now director of engineering at Facebook, put it, “AWS is built around a model of sharing resources; hardware, network, storage, etc. Co-tenancy [that is, sharing services] can introduce variance in throughput at any level of the stack. You’ve got to either be willing to abandon any specific subtask, or manage your resources within AWS to avoid co-tenancy where you must. Your best bet is to build your systems to expect and accommodate failure at any level.”5

The continuing software development dialogue with manufacturing
The theory and practice of organizational performance improvement may be going full circle—from every product built slightly differently to everything highly standardized and then back to high product variability. But this time, the variability is not accidental and loss producing; it is intentional and value creating, enabled by more and more granular flows of information.

Frederick Taylor’s “scientific management” transformed craft production into mass production by surfacing human knowledge and placing it into documentation,

---

processes, and tools. Command and control management techniques imposed standards from the top down—epitomized by the notion that “You can have any color car you like as long as it’s black.” (See Figure 4.)

W. Edwards Deming recognized that quality and value were diminished when information flowed only one way, and he taught the Japanese and then others about quality circles and plan-do-check-act (PDCA). He understood that feedback from shop floor employees was key—but in service to mass production of standard products. Mass personalization at the product level via highly programmable robotics now makes possible one-off car purchasing options—creating the appearance of a car built by hand “just for you,” as a craftsperson might have done in 1890.

Enterprise software has this same life cycle, but hasn’t always taken stock of Deming’s teachings. Enterprise applications have been very effective at standardizing information capture and distribution through enterprise resource planning (ERP), customer relationship management (CRM), and other packages. But standardization has meant a repeat of the top-down, one-way information flow Deming decried.

The web, with its direct connection to customers, raised the value of personalization and, more importantly, the need for feedback. Agile software development methods have answered that call. Now the more complete digitization of work—from back office to front office to supply chains to distribution channels and even customer consumption—faces the challenge of dynamic markets where change is constant. Leading companies are responding by running software in the cloud and adopting DevOps methods to support the continuous delivery of digital responses to market shifts.

**Not moving toward antifragility—yet**

According to VersionOne’s latest annual survey, 48 percent of global enterprises have five or more active agile development teams.⁶ There isn’t much evidence yet that they’ve moved toward antifragility or even know how to do so. But the concept does make sense to some who see it on the horizon and “get” how it might become part of their current agile development initiatives.

For example, Gopal of Nationwide says, “It just scares me to think about somebody unleashing Chaos Monkey within Nationwide, but we are taking somewhat of an intermediate step. We want to have more rigorous discipline around high-availability testing, and not just when completing major projects.”

Given the nature of Nationwide’s industry, viewing technology change through a risk lens is appropriate, and the company sets goals based on a rolling risk assessment. Nationwide realizes that a new definition of stability is emerging, but hasn’t seen a feasible way to reposition itself in light of that realization yet. Gopal notes three main categories of technology risk:

- Technical debt, such as a compromise made to meet delivery deadlines
- Currency, such as deferring the cost of upgrades
- Consumption management (or efficiency from the vantage point of supply levels)

The methods associated with antifragility could have an impact on all three categories.

---

Even before the executives interviewed for this issue of the Technology Forecast encountered the antifragile concept, agile development has been a primary source of inspiration for them. Agile methods have been used for more than a decade, so the connection between practical business goals and agile methods is quite clear at this point. For some, the goal is efficiency.

John Esser, director of engineering productivity and agile development at Ancestry.com, for example, says efficient infrastructure is critical for the 23-year-old company. “In our case, it would take as much as a couple of months to get new services provisioned and set up,” Esser says. “We didn't have the flexibility to move resources around. Existing hardware was dedicated to current services.” Esser points out that although the development process was lengthy, “the big long pull was that of our IT infrastructure.”

At Ancestry, better configuration management via automation has been central to reducing the time it takes to provision a new service. “We definitely improved the development side, and we definitely saw increases and productivity gains there, but we could get only so far until we addressed the infrastructure issues,” Esser says.

Like Gopal, Esser tracks developments at major web companies. “I've been inspired by other companies, obviously Amazon and the whole idea of the infrastructure as a service and the cloud. Netflix is another one that I've looked toward when it comes to some of their processes. In fact, we've been exchanging ideas. In its day-to-day engineering, Netflix embodies a lot of what we're trying to aspire to. Etsy is another good example of just pure scale. And Google is a great example for me, because the company has done such a great job of standardizing its infrastructure in a way it can leverage very quickly.”

Conclusion: Balancing quality, speed, collaboration, and resilience

These enterprise interviewees confirmed that they are on a journey of sorts—a quest to improve how they operate. None asserted that their development or other process improvement efforts are where they need to be yet. The antifragile concept is new to these executives, but it makes sense. Agile methods in businesses are forcing changes in IT, leading to the adoption of change management techniques in IT similar to the antifragile approach web-scale companies use to manage change.

As companies benefit from agile methods for software development and from watching what leading cloud development efforts can achieve, they are beginning to consider the possibility of using antifragility principles in support of broader organizational change. But these and other enterprises don’t view agile methods as a panacea, nor should they. During the last 80 years of building things, the emphasis has shifted from quality, to speed and efficiency, to resilience and stability, to scalability. Now, with web-scale companies beginning to work toward antifragility, the business world is returning to considerations of quality.
Reconsidering risk in a continuous delivery environment

Continuous Delivery author Jez Humble says culture, rather than risk, is the real obstacle to transforming IT.

Interview conducted by Alan Morrison, Bo Parker, Bud Mathaisel, and Robert L. Scheier

PwC: Many CIOs are accustomed to buying packaged software or subscribing to services on the web, and they may even outsource the development that must be done. They may not be mentally engaged with the issues you explore in Continuous Delivery. Is there an argument for these CIOs of established enterprises to reengage?

JH: You still often see the same problems with packaged and COTS [commercial off-the-shelf] software, which people buy because they don't want to do custom software development. They see that as risky.

They think COTS is going to handle a particular business process, and then they find their process does not match what the COTS package does. So they decide they need to customize the COTS. They end up doing custom software development to customize the packages they bought. And packaged software is not normally designed for that.

As a result, they pay a lot of money for a consultancy to heavily customize the software. That's extremely painful, and upgrading the software also becomes extremely painful. Often, the vendors that sell the packages will not support heavily customized versions of the software. We see this all the time in enterprises.
PwC: Do we need a whole risk reassessment, and do we need to revisit some preconceived notions about where the risk actually lies? A CIO might have a risk concern about open source, for example. Where do you place that risk? JH: In our experience, enterprises are very aware of that risk. But they don’t seem to be quite so aware about the risk of many of their legacy systems already in place. I’ve given talks and consulted extensively in enterprises. And here is one of the standard questions I ask: How many people are running mission-critical systems on legacy software or on legacy packages, where the vendors no longer support the packages or the hardware they’re running on, and where you don’t have the source code to the customizations? People don’t want to touch those legacy systems, because they worry about breaking them. But the more you don’t touch them, the more the organization loses the muscle memory associated with being able to manage those systems over time. And then it gets more and more risky.

One of the things I read this past year that has really inspired me is [Nassim Taleb’s] book *Antifragile*, in which he talks about the behavior of systems and how they respond to unexpected events. Like financial markets, legacy systems represent a huge risk to organizations. Inasmuch as organizations do need to change those systems eventually, it’s very risky. I think organizations always need to look at how to mitigate that risk by making sure they’re up to date with current software and current trends and what’s supported in the market.

Open source actually provides a great way to mitigate some of those risks, because you have systems, packages, libraries, and software where the source code is freely available and many eyeballs have been on that code. You have a large community of people who have the knowledge to use those systems and support them over time. Companies have problems hiring people to support or maintain the systems that they’re running in production. They don’t have the source code anymore. A lot of these open source code tools have large communities of people. So it’s much easier to hire people with experience, and it’s much easier to maintain the tools because there’s a large community built up around them. Vendors are going out of business or being acquired at an ever higher rate. For enterprises, that is a much higher risk to the software they’re buying than open source.

PwC: What are the points of friction between the DevOps or continuous delivery approaches and other approaches such as ITIL [Information Technology Infrastructure Library—a collection of best practices for IT service management] that have been around for years? JH: There doesn’t need to be a conflict. I have the ITIL Foundation certification. I’ve studied the framework extensively, and based on my study, ITIL is compatible with agile and continuous delivery. The main point of friction is often the change management process, but ITIL provides several mechanisms for lightweight change management processes, which are completely compatible with continuous delivery.

ITIL is a library of good practices that should be adapted by the organizations that implement them. But frequently that is not the case on the ground. Some people have come in with a very strict, heavyweight implementation of ITIL and said, “This is what you have to do.” And that’s neither in the spirit of ITIL nor is it the only possible way to implement ITIL. Unfortunately, a lot of companies have implemented very heavyweight practices that actually increase the risk of things such as downtime and lack of stability in production.
“People are very nervous about the security of public cloud, but time and time again we see hackers getting into company data centers, including potentially government-sponsored groups.”

**PwC: To use Geoffrey Moore’s terms, the development model for systems of engagement is entirely different from the development model for systems of record and the core IT. Systems of engagement are often heavily dependent on the public cloud, which is another set of risks. How do you see systems of engagement in this risk environment that we’re talking about?**

**JH:** I think public cloud has made it a lot easier to respond to the market. Software as a service, infrastructure as a service, and platform as a service—these systems pose risks that are different from traditional enterprise IT risks. Those risks include issues such as security and, in particular, making sure you don’t put data in the cloud that could then be compromised.

There are also architectural issues. Building software for the cloud requires architectural approaches completely different from traditional IT approaches. For example, you need the ability to scale horizontally so you can deal with volatility, such as making sure your software doesn’t go down if one data center goes down—which has happened several times recently.

People tend to be very nervous about the risks of the new technologies, and they tend to be somewhat blind to the risks posed by their existing platforms. For example, people are very nervous about the security of public cloud, but time and time again we see hackers getting into company data centers, including potentially government-sponsored groups.

It’s very easy to be skeptical of this new thing and be very nervous, but it’s just a different set of risks. It’s not a worse set of risks. The business is going to force these people to get up to speed. A lot of this skepticism is driven by fear and nervousness about change, which is understandable. But that’s not going to be addressed by people putting their heads in the sand.

**PwC: The collaborative development cycle that happens on the open web is very powerful. How are enterprises approaching open collaboration? Are there certain kinds of development they can do in the open?**

**JH:** Some companies are starting to do so. I think things inevitably will move in that direction, because it’s the only way to move sufficiently fast to address the business needs.

Companies such as Etsy are driving a competitive advantage from being able to do open collaboration. Etsy is not a small company. The company has put a lot of work into open sourcing a lot of its infrastructure. Facebook is open sourcing a bunch of its stuff.

Trying to hire people who can build software in this way is really hard, and being able to engage with an open community and prove that you’re contributing back is a really powerful tool in trying to hire in skills.

An article in *Forbes* a year or two ago described how every company is a software company. That’s the shift I’m observing. How do you actually execute on that? The biggest problem is the Taylorian management paradigm of the 19th century, which is still very prevalent in our industry: You can drive out a very detailed specification and then throw it over the wall to IT. A bunch of worker bees do the boring and trivial work of writing the code, and then they toss that over the wall to operations to run to eternity.

That model of building software is not going to work. It’s impossible to predict in advance how customers will respond. Building a really large, complex specification is not a good way to mitigate the risk that the software may not deliver a return on an investment.

Companies need to understand that. The main problem is hiring the people who know the details of the continuous delivery methodology. They are in short supply, and they’re not going to work in a company that has a traditional software factory approach. Companies need to take a very lean, startup kind of approach to building development teams—try some things and experiment.
PwC: What’s the biggest obstacle to moving to this more experimental approach?
JH: The biggest problem is culture. Organizations need to take a cultural approach in which they experiment, try things out, and manage risks by not investing too much up front and by enabling the team to experiment and innovate. You can be very incremental about growing those skills, and then you can gradually seed them out through your organization and grow the rest of your organization in that way.

The lean startup is something that Intuit has been known to adopt, for example. Intuit is not a small company. There are proven models for adopting these approaches. You need to focus on creating a culture of learning and experimentation and being very incremental in what you roll out.

We’re in an age where the life of the Fortune 500 company is steadily shrinking. Continuous evolution should be a part of every company culture. This is a journey. We’ve taken step one in the journey, and we will need to try some other things as well, as we continue to evolve.
How established enterprises are moving toward DevOps

Vijay Gopal of Nationwide Insurance discusses DevOps within the context of an insurance company’s view of risk.

Interview conducted by Alan Morrison, Bo Parker, Bud Mathaisel, and Robert L. Scheier

**Vijay Gopal**
Vijay Gopal is a vice president and enterprise chief architect at Nationwide Insurance.

---

*PwC: What is the development philosophy at Nationwide, and how is it evolving?*

**VG:** The biggest characteristic of our development philosophy is that we embrace the agile methodology. At our Application Development Center, we have very good predictability and quality, and we hope to harvest some of those benefits and get more of the organization to embrace the agile methodology.

Our stated goal is to have 50 percent of all our development go through the Application Development Center by the end of the year. The processes of test-driven development and involving the business as part of the development are both catching on and proving to be very successful.

But when it comes to the cloud-based development paradigm, it’s a little harder to gain a foothold. Our organization is just getting their heads wrapped around agile. We perform more than $400 million worth of build work every year, so we must be able to do that development at scale. We are one of the few CMMI [Capability Maturity Model Integration] level three certified shops that also embraces agile practices, and we also must be mindful of organizational change management as it relates to how these systems are being deployed to our user groups.

We see some challenges in embracing the cloud-based development. That said, we are experimenting with it for our internal applications. We have an area
called enterprise applications, and we are developing an application catalog. To build this catalog, our developers take the functions that associates frequently access from our intranet portal, and then our developers create easily accessible widgets. Instead of going through five or six clicks to find out how much time off they have, for example, associates could click on a widget and get that information quickly.

We are becoming better informed about what it takes to do that as a company, while still getting the organization to shift and embrace the agile principles more effectively.

**PwC: Are you able to do more frequent releases?**

**VG:** We’ve had a monthly release calendar for the past four years. Before that, each of 26 business units had their own release schedule, which was chaotic for users. Going to a monthly release schedule provided some discipline for managing all the defects that get introduced because of change. We want to take a closer look at how companies such as Yammer or Salesforce.com do daily or weekly releases while keeping the user experience from being disrupted.

We need to have some discipline when we roll change out to the call center associates or the agencies, for example. When we introduce change into these environments, we must be very careful about the impact it will have on people who have been through a training program, have used some of the features for a long period of time, and are accustomed to an approach where the field associates deliver some training to orient them to those upcoming changes.

It’s the other side of the spectrum from the world of the web companies, where code once validated gets released. Those applications are intuitive enough that the users don’t need to be trained to take advantage of the features.

The way I see Nationwide moving toward more frequent releases is by learning from all the SaaS [software-as-a-service] solutions that we are starting to embrace, and from the packaged implementations that we’re bringing into our environment more and more.

**PwC: Could you tell us a little bit more about how you’ve been able to bridge the agile approach and the working code approach in a package world?**

**VG:** Even when it comes to legacy modernization, such as when we implement a packaged solution to replace our core policy administration systems, we take an agile approach. We put a heavy emphasis on working code rather than project status reports, for example. From a DevOps standpoint, we are rolling out more and more of the private cloud solutions, especially in a Java environment.

When going the package route, you must make some conscious decisions about how much customization you will take on. Our philosophy has been that the package implementation itself will be successful if you also make some business process changes that help you take advantage of the package implementation.

We get a feel for how the application will behave. And to go along with the package implementation, we also have a business effort called product simplification. Our product varies by state to reflect the differences between how marriage is defined in one state as opposed to another, for example. That’s an opportunity to give more flexibility to our field offices.
That’s a huge change for our business, but the field offices had to sign up for that so the organization could see all the benefits that come with the package application. Now once we have done that, the question becomes one of the release schedule and clarifying whether we’re taking a state-based or a product-based release schedule.

We started with the simplest products first. We make accommodations for the business rules that need to be put in place and any minimum customization required to make that happen. That becomes the first iteration deliverable. When the updates come from the package supplier, these principles will help protect us from being disrupted by an upgrade.

PwC: Can you talk a little bit about where your operations shop is in terms of its ITIL [Information Technology Infrastructure Library] adoption and whether or not you see any tension between agile and ITIL?

VG: We definitely see some tension. ITIL adoption has been mature in a couple of disciplines, such as incident and problem management, and we’ve realized some benefits from adopting these practices. The tension between the agile and ITIL practices is somewhat mitigated by our monthly release calendar. The teams understand that all our iterations in the test center are two-week iterations, and we’ve structured our work within that monthly release calendar.

PwC: Do you think you may move more toward DevOps in the future?

VG: I think we are more likely to go toward DevOps, but before we get there, we need a standardized infrastructure environment. While we’ve embraced virtualization, there’s more of a craftsman mentality in how Java fills the solution. To deploy and manage infrastructure in the scale at which we consume it, we need to rationalize workload patterns with our development teams.

PwC: Once you’re comfortable with those, then you’ll start looking to do more of a full life cycle, DevOps-style approach. Is that a good way of thinking about it?

VG: That’s absolutely how we are approaching it. We’ve finally reached a point where we cannot defer the basic hygiene work that we need to do around systems anymore, in terms of the technical debt and taking care of some of the risks we have in our systems. Some of that was deferred in favor of rolling out more business capability.

PwC: Have you been able to explain this concept of technical debt to business leadership?

VG: Yes. A couple of years ago, we consolidated the amount of technical risk we have in the system. The first step was to get visibility about how much risk we had in the system and finish that step in time for our annual planning cycle. A large business transformation committee makes all the decisions on allocating funds for projects.
We highlighted any significant risk, provided an estimate, and proposed that we invest to mitigate the risk. We hoped to avoid a situation in which one business unit spends a significant amount of money because they’ve made a good case while another business unit has much higher risks that are not being tended to because a business capability is being rolled out.

This year we’ve discussed baking into our operational budgets the mitigation of technical risk or taking care of technical debt. When people run budgets, they tend to assume that they can base their budget for next year on the money they got this year. We’re approaching technical debt as an item that will be added to every year, so we assemble the risks, prioritize them, figure out what needs to be fixed, figure out the level of effort required from each of the business solution areas, and explicitly add that amount to their budget. It’s a little more of a disciplined approach for shared risk on platforms that are used by more than one line of business.

We have three broad categories that we bring under technology risk. The first is technical debt, such as a compromise made to meet project delivery deadlines. The second is currency, such as deferring upgrades to an environment because of the cost associated with it. The third is consumption management, or efficiency, where making changes will help you optimize how much users consume. All of those we put down as risk, and we manage them accordingly.

“We would need to prove this agile development in some of the noncritical areas of our business first, before we could start talking to our business about this kind of an approach.”
Microfilm digitization station at Ancestry.com in Provo, Utah
Making DevOps and continuous delivery a reality

Automation, thoughtful choices, and a well-designed workflow are pivotal to making good progress.

By Robert L. Scheier and Alan Morrison

When Chad DePue, founder of Inaka, launched his 30-person software development and hosting company, he was determined to apply the lessons he had learned from standard enterprise software development practices—lessons about what to avoid.

During an earlier period as vice president of product and program management at a secure e-mail and workflow vendor, DePue observed the “tedious, laborious, and slow” way that most enterprises roll out technology infrastructure and applications. For example, stacking hardware in a company-owned data center and plodding through ponderous cycles of as long as one year to develop, test, and deploy each major code release.

“That’s not an environment I find very exciting as a developer,” he says. It certainly does not attract today’s most skilled and innovative developers, whose work is in high demand by customers. Nor is it conducive to the market demands on Inaka.

At four-year-old Inaka, based in Buenos Aires, DePue is creating what PwC calls a more “antifragile” enterprise, an organization that improves and gets stronger as a result of unpredictable change and environmental volatility.

A DevOps, continuous delivery approach provides a stepping stone to that antifragile state. At one level, DevOps is a working style designed to encourage closer collaboration between developers and operations people: Dev+Ops=DevOps. Historically those groups have been working more or less at cross-purposes. DevOps collaboration seeks to reduce the friction between the two groups by addressing the root causes of the friction, making it possible to increase the volume and flow of production code, to improve the efficiency of developer teams, and to reduce the alienation of the ops people who must guard the stability of the system.

One of those root causes of friction is poor workflow design. DevOps encourages extensive automation and workflow redesign so developers can release small bits of code frequently (in a more or less continuous delivery cycle) and yet not disrupt the operational environment in doing so. The workflow includes buffers, compartmentalization, and extensive monitoring and testing—a very extensive and well-designed
DevOps encourages extensive automation and workflow redesign so developers can release small bits of code frequently, and yet not disrupt the operational environment in doing so.

Pipeline, but also a rich feedback loop. It’s a very test-driven environment. When the small bits of code get deployed, the individual changes to the user experience tend to be minor or limited to a small subaudience initially.

Inaka’s cloud-based service offerings are never finished. In the cycle of collaborative development, deployment, testing, and revision, the services are repeatedly reinvented through ongoing and fast-paced experimentation, deployment, and tweaking. The services are more stable because of the pace of that reinvention. When something breaks, it is fixed within a short period of time, and when something is improved, that improvement arrives quickly.

To match this continuous delivery style of today’s best developers and to meet ever-changing market demands, DePue relied heavily on open source software. “Inaka’s main goal is to be the new outsourcing stack,” he says.

It might be tempting to dismiss Inaka as a greenfield example, but the demands on it are similar to those on larger, older, and more traditional businesses: fast-changing technology and customer requirements, plus unrelenting pressure to cut costs and speed delivery. Although it does not face the pressures of operating in a heavily regulated industry, Inaka is a serious competitor in its field. Clients range from web startups to Fox News Network and Viacom, VH1, and MTV. One app it developed and hosts, Whisper (a pseudo-anonymous social network for college students), made the top 100 in the Apple App Store.

Startups and established enterprises alike are using open source software to become antifragile organizations capable of meeting unpredictable business needs more quickly and inexpensively through continuous delivery supported by the tools described in this article.
From the smallest social media startup to the oldest traditional financial exchange or manufacturer, the business environment is changing far too quickly for the software deployment cycles of the past to be effective. When a new mobile or social platform surges into popularity, or a natural resources boom creates an emerging market, enterprises must quickly tap that potential with new products and applications.

Increasingly, these enterprises will need to become more antifragile, as explained in the article, “The evolution from lean and agile to antifragile,” on page 06. For the IT department, that means adopting a continuous delivery software stack to enable rapid, collaborative, and massively scalable application deployment, testing, and monitoring in an ongoing and increasingly automated process.

Neither open source approaches nor agile development methods are new. But standalone open source tools and entire frameworks for application development and deployment are new. Also new is the extent to which these tools are being adopted along with the related process of agile development to bring more speed, flexibility, and collaboration to enterprise IT.

For example, GitHub (the commercial version of the open source Git, a version control system) now has more than 2.7 million users and 4.6 million repositories. (See Figure 1.) More details about Git and GitHub can be found later in this article in the section “Requirement 2: Collaborative, iterative development.”

In these continuous delivery environments, the organization’s processes and tools must support a deploy-and-fail-fast mindset that constantly reacts to market changes. The tools in this stack must do the following:

- Provide not just point capabilities but frameworks that encompass multiple parts of the continuous delivery cycle
- Interact and share information to foster real-time responsiveness and collaboration across business and geographic boundaries
- Automate manual functions wherever possible to increase speed and reduce costs
- Support rapid, iterative, and continuous development, deployment, and testing

In this “new age” IT, what previously was a physical infrastructure is more a virtual entity to be automatically deployed, tested, reconfigured, and reused through scripts. The traditional long, complex sequence of gathering requirements, developing code, testing, deployment, and monitoring becomes a collaborative, iterative, and never-ending process.

Commercial and open source vendors offer tools to automate, integrate, and increase the speed of each function. However, the market is young, and many of these tools are still immature. Veterans warn that IT organizations may need to write some of their own tools for the open source IT stack.

From the smallest social media startup to the oldest traditional financial exchange or manufacturer, the business environment is changing far too quickly for the software deployment cycles of the past to be effective.
own tools or adopt different tools for different needs. More cautious organizations may also want to keep some work manual until they fully trust automation or can see a suitable return on their development effort.

The rest of this article examines the main requirements for continuous delivery and some of the tools available.

**Requirement 1: Become antifragile**

Conventional planning for robust systems usually involves avoiding equipment failure at all cost, buying the most expensive (and supposedly most reliable) components, and clustering them so a backup can take over for a failed component. An antifragile approach requires a continuous delivery software stack that turns these traditional processes upside down.

As practiced in the open source community, antifragile thinking assumes that systems comprise large numbers of lower-cost commodity hardware nodes, some of which are bound to fail unpredictably. An antifragile approach also can involve deliberately causing unpredictable failures in components within a cloud infrastructure to detect and remove vulnerabilities.

Some of these failure-causing tools, once used only by cloud-based service providers, are now available as open source software. They include Netflix Chaos Monkey and Stelligent Havoc, which randomly shut down Amazon EC2 instances to help developers plan for such disruptions and fix weak points in their infrastructures.

To cope with the expected failures these tools uncover, organizations can use clustering and other traditional measures. But organizations also must use newer techniques, such as:

- Providing a scaled-down but adequate version of a service in case of a component failure. For example, if an e-commerce site suddenly lacks the compute power to provide personalized choices for each customer, it could still provide a list of the most popular choices for all customers that require less processing.

- Removing as many dependencies as possible so the failure of one component has the smallest possible impact.

- Creating workarounds that keep the enterprise operating even if services once thought essential are no longer available. These workarounds include ensuring that traffic load balancers correctly detect and route requests around instances that go offline, reliably rebuilding instances, and ensuring that patches made to one instance are committed to the source repository. Continuous delivery often allows rapid fixes.

**Requirement 2: Collaborative, iterative development**

Web-based, open source version control has transformed what once was the routine housekeeping function of storing code under development. Such version control is now the center of dynamic and creative development environments. Traditional source code repositories commonly stored work from internal developers who wrote proprietary apps using commercial integrated development environments. Today’s web-based repositories let developers around the world develop and share multiple branches of code, reuse code, and easily combine branches. They also make it easier for developers to share code libraries for commonly used functions, reducing the cost and time required to bring new services to market.

GitHub, the commercial version of the open source Git developed by Linux spearhead Linus Torvalds, is one of the hottest open source version control systems. Part of its appeal is its social features, which promote collaboration, and with more than

Commercial and open source vendors offer tools to automate, integrate, and increase the speed of each function. However, the market is young, and many of these tools are still immature.
3.5 million users, GitHub is gaining share on Apache Subversion, a more centralized version control system.

“This is the language of open source now, and it’s all about source control and how you manage changes to code,” says Charles Oppenheimer, founder of Prizm, which has developed the MightBuy shopping app. “If you aren’t coding this way, you aren’t speaking the new language. I don’t hire anybody who doesn’t have GitHub contributions, so I can see their code.”

GitHub Inc. has launched GitHub Enterprise, a version of Git for an organization’s internal network, which is protected by the customer’s existing authentication. Stash from Atlassian also provides behind-the-firewall Git repository management for enterprise teams. Microsoft has announced support for Git in its own Team Foundation Server (TFS) version control system and in its Visual Studio development tool.

**Requirement 3: Rapid coding**

Speed and simplicity are essential when business models and delivery channels (such as mobile) change constantly. This necessity requires programming models that expect data to be presented on different devices and that promote code reuse across platforms.

One of the most popular is JavaScript. Its support for user-defined functions allows the development of a library of reusable code that can be pasted into future applications and called through a statement. Rated most popular in a first quarter 2013 ranking of development languages according to activity on GitHub and the Stack Overflow developer discussion site, JavaScript is widely used to speed the performance of web pages and to embed capabilities that make websites more interactive and compelling.1

A popular tool that complements JavaScript is Joyent’s Node.js, which is server-side software used to create scalable Internet applications, minimize overhead, and maximize scalability while allowing developers to create the server and client side of an application using JavaScript. A major retailer, for example, chose Node.js for mobile application development because of its scalability and ability to communicate with other services, such as the retailer’s application programming interface (API) and database.

Another challenge is linking modern web-based or mobile applications to existing databases, transaction processing systems, and other legacy services. To ease such connections, eBay released ql.io, a data-retrieval and aggregation gateway for HTTP APIs.

Apache Cordova (formerly PhoneGap) is a set of JavaScript libraries that developers can invoke to run device-specific native code for each platform and utilize native device capabilities such as the accelerometer, camera, or contact list.

When developers combine Cordova with a user interface framework, such as jQuery Mobile, Dojo Mobile, or Sencha Touch, developers can use HTML, CSS, JavaScript, or other familiar web languages to create apps for multiple platforms, rather than using separate native code for each platform. Cordova also bundles the application assets into a package ready for distribution through each platform’s app store. Disney World, Stanford University, and SlideShare are among the many organizations that have used the HTML5-based jQuery Mobile to build mobile applications to run on multiple platforms.

The open source IT stack must also be built for high performance and scalability. Erlang, an open source programming language whose runtime system supports concurrency, distribution, and fault tolerance, is designed for massively scalable real-time systems that have high-availability requirements. It is used in telecom, messaging, banking, finance, gaming, and embedded systems.

Today’s web-based repositories let developers around the world develop and share multiple branches of code, reuse code, and easily combine branches.
**Requirement 4: Automated, rapid configuration and deployment**

Users chronically complain about IT’s inability to rapidly configure servers, applications, and other resources. For that reason, the ability to instantly—and even automatically—create infrastructures from scripts is one of the key goals of continuous delivery.

“Dependency management is a very hard problem to solve,” says a software development manager at a major smartphone handset vendor. “If we change one component or the internals of one of the components, we need to figure out the effect that would have on every other component.”

Rather than rely on complex dependency management software, his organization uses a manual process of releasing one changed component at a time to a “near live” environment and then putting it in production only if it passes rigorous tests. He trusts this manual process, discipline, and “the fact that our developers know what they are doing.” For a larger organization that has more services to track, he says, a deployment tool probably would make better sense. These tools now are more common and more advanced than when he began the move toward continuous delivery, he notes.

Open source automated configuration tools, such as Puppet and Chef, replace older, slower methods, such as structured scripting and image cloning. Rather than rely on “golden images,” these tools allow organizations to express their infrastructure as code and to share configurations through GitHub or other version control systems.

Being first to market, Puppet has wider platform support and more users than Chef. System administrators often find Puppet’s model-driven approach easier to use because it relies on data structures created with the JSON (JavaScript Object Notation) data interchange format. Chef’s procedural approach, which requires writing configuration “recipes” in Ruby, often makes it a better match for developers.

Inaka uses Puppet and Chef to automatically and quickly provision servers “depending on the application’s needs,” DePue says. “The closest we get to a physical server are companies like Liquid Web,” which custom provisions servers similar to Amazon Web Services servers so DePue’s customers get dedicated higher-performance hardware at the price of lower-performing public-cloud multitenant servers.

The need to train users in Chef and to maintain those skills over time can be a risk, but the tradeoff is increased power and flexibility through the use of its command-line interface, called Knife. At first glance, Puppet’s model-driven approach means less fine-grained configuration control, but the latest version also allows developers to craft configurations using Ruby.

John Esser, director of engineering productivity and agile development at Ancestry.com, has been using Chef in a three-year effort to move the online research site toward agile development and more frequent code deployments. Before Chef, he says, “even replicating a given server’s configuration was very challenging, if not impossible.”

He likes Chef because of its Windows support and its use of Ruby rather than a customer-defined domain-specific language. He can easily create code to handle unique configuration requirements.
Along with Chef, Esser is using Go, an agile release management tool from ThoughtWorks Studios, as a “continuous delivery dashboard that indicates the status of development and lets the team push the button when things are deemed ready” to deploy. These tools help his developers and operations staff to leave their traditional sequential approach and adopt continuous fail-fast deployment, enforcing standards “through tooling and automation,” he says.

Niek Bartholomeus, an independent software architect in Antwerp who recently helped a European bank move toward more frequent delivery, says Chef and Puppet “are declarative in the sense that you give them a desired end state and they will make sure the infrastructure is aligned properly.” He likes the idea of using such tools to automate the provisioning of operating systems and middleware. For the bank, however, he chose StreamStep’s SmartRelease (since acquired by BMC Software and renamed Release Process Management) because it requires either notification of a staff member or a custom-developed script before deployment. “Chef or Puppet would be too big of a leap in one step” from the bank’s formerly highly manual application deployment processes, he says.

Others have chosen configuration tools from major vendors. For example, Vijay Gopal, vice president and enterprise chief architect at Nationwide Insurance, uses a configuration platform from Hewlett-Packard.

**Requirement 5: Ongoing, automated, and proactive testing**

When speed is imperative, testing is a necessary evil—but a very necessary evil. In a 24/7 social world, and especially for web-facing applications, a slow or failed site or application can sink a business.

In a 24/7 social world, and especially for web-facing applications, a slow or failed site or application can sink a business.

Continuous delivery tools aim to meet this challenge not only by automating much of the test process, but also by turning conventional test methodologies on their head. Rather than making test a separate and lengthy sequence in the larger deployment process, continuous delivery practitioners roll out small upgrades almost constantly, measure their performance, and quickly roll them back as needed.

Automated testing tools include the Ruby-based Cucumber, which allows development teams to describe the desired behavior of an application in a business-readable, domain-specific language. For unit testing, QUnit is a framework for testing any generic JavaScript code. JUnit is a framework for writing repeatable tests for Java applications.

Twist, a tool from ThoughtWorks Studios, assists development teams in creating and maintaining test suites that serve as a bridge from manual to automated testing. CruiseControl is also used as a continuous integration tool and extensible framework to integrate the continuous building and testing of software projects.
Inaka uses the Keep It Functional (KIF) open source test framework from Square for iOS apps, JUnit for testing Android apps, and Erlang’s Common Test framework built into Ruby.

At Ancestry.com, Esser avoids automated test tools, especially those that record and play back interactions with an application. He sees them as “not conducive to agile.” Instead, he uses test engineers that “move from team to team to help the developers build and own their testing.”

Ancestry is just one of several companies interviewed who are moving beyond agile by placing more emphasis on a test-driven, continuous change environment. (See Figure 2.)

**Requirement 6: Real-time monitoring and performance management**

Through monitoring and performance management, administrators receive and act on real-time feedback to be sure each new function works properly and to release new code that fixes anything from performance to usability issues.

Some deployment tools, such as Jenkins, use plug-ins to monitor the executions of repeated jobs—for example, building a software project or jobs run by cron and other schedulers. Jenkins can also monitor the execution of job runs on remote machines, capturing their output for analysis by administrators.
Boundary, an application performance-monitoring provider, uses a software-as-a-service (SaaS) delivery model. Other commercial tools include Nagios and Librato, Amazon CloudWatch (provided for and by the AWS environment), and Splunk. Open source tools include Graphite, Kibana, logstash, and more.

Says Inaka’s DePue, “For the physical monitoring—‘Are the servers OK?’—we’re using the Nagios monitoring suite,” which provides basic data for all servers on an in-house dashboard. To track the “missing piece” of server and network performance across the cloud, Inaka uses Boundary.

A real-time dashboard and visualization capabilities drew DePue to Boundary. “We rolled out our system for a client using a popular open source database, but we had some real scalability problems with the database and couldn’t figure out what was going on. We finally took some screen shots of the graph of network usage and immediately saw that our network interface cards couldn’t handle the load, as traffic across the NIC would collapse and then slowly recover,” DePue says.

When Boundary revealed that some programming Inaka hosted for a customer was running up excessive network charges, DePue moved it to CloudFlare, a content delivery network. CloudFlare routes customer traffic through its global network, optimizing delivery for the fastest load times and best performance while blocking threats. While the basic CloudFlare services are free, additional offerings—such as faster performance, 100 percent uptime guarantees, and advanced security protection—incur a fee.

“CloudFlare is like an insurance policy, where you pay them only if somebody attacks your service or you suddenly need more capacity,” DePue says. By dynamically assigning more servers to counter the attack, “they help you weather the storm. Ten years ago, you would have needed to buy a bunch of extra hardware for that protection.”

Splunk complements Boundary’s network monitoring SaaS application, says Cliff Moon, Boundary’s founder and CTO: “Splunk focuses heavily on log-based data and has a great analysis engine.”

Esser of Ancestry.com notes, “We view monitoring at two levels: the hardware/virtualization system level, such as CPU, memory, throughput, etc., and the business cluster level—that is, each key service is in a cluster that must meet a service level agreement. At the hardware/virtualization level, we use the monitoring tools provided by Microsoft HyperV. At the business cluster level, we have a custom-built solution that integrates and monitors each business level cluster.”

Unified tools for unified work
Properly used, the new generation of automation and scripting tools described in this article are turning app development, deployment, testing, and management from a series of discrete, cumbersome events into a nimble recurring process that speeds business agility and success in fast-changing markets—and that supports more antifragile organizations.

However, the market is still young. “All the capabilities needed for a continuous delivery stack are largely immature,” Esser warns. “You have a few gems like Chef, Go, Puppet, or others, but organizations are left to build their own solutions or piece together various tools.” The choice of tool also depends on scale. “Large websites like Ancestry.com, Amazon, and Netflix employ scaling techniques different from those used by smaller organizations,” he says. “It also depends on your core technology. Linux is more mature and offers more open source solutions, while Windows is a tough road.”
Monitoring is “still a very thorny problem,” says Jez Humble, a principal at ThoughtWorks Studios. “A lot of companies have built their own stuff to do this. We’re seeing a whole new area [of] application release automation emerge,” Humble says, but most of the tools work only in greenfield environments—not with the legacy applications that exist in most established organizations.

The best tools mirror the ongoing and parallel development, test, and deployment activities at the core of continuous delivery. “Look for something that integrates the information that comes from various functions, such as change management, version control, issue tracking, and monitoring, and that can be used by developers, operators, testers, and others,” Bartholomeus says. “For me, that’s an ideal start for one team to get insight in the other team’s work and to help bridge the gap between them.”

**Conclusion: Start now**

Whatever an enterprise’s size, age, or market, today’s ever-changing business requirements demand a continuous delivery software stack that enables rapid, collaborative, and massively scalable application deployment. The organization’s tools and processes must support a deploy-and-fail-fast antifragile mindset that constantly reacts to market changes.

Rather than wait for commercial software vendors to meet all the tool needs, open source developers are building on each other’s work to repeatedly refine tools across the continuous delivery spectrum. This open source stack manages a virtual, cloud-based infrastructure that is automatically deployed, tested, reconfigured, and reused through the use of scripts.

Not every enterprise needs—or is ready—to move completely to a continuous delivery stack and a continuous delivery way of working. Some legacy apps and systems of record may not be suited to the cloud. Some highly regulated industries may not be comfortable fully automating the deployment of sensitive applications. Some corporate cultures may not be comfortable immediately abandoning long-standing habits of separate requirements, development, testing, and deployment cycles.

But the urgent business needs that drive continuous delivery—speed, quality, efficiency, agility, reuse, and low cost—are intensifying. CIOs should educate themselves and their staffs in this new IT mindset and the tool stack that enables it, so they’re ready to move their organizations toward a continuous delivery culture when—not if—it becomes urgent.

**The urgent business needs that drive continuous delivery—speed, quality, efficiency, agility, reuse, and low cost—are intensifying.**
Using DevOps and continuous delivery to move at the pace of the business

John Esser of Ancestry.com discusses his role in the company’s transformation efforts.

Interview conducted by Alan Morrison, Bo Parker, and Bud Mathaisel

John Esser

John Esser is director of engineering productivity and agile development at Ancestry.com.

PwC: How did you get started at Ancestry?
JE: I’ve been here at Ancestry about three years. I was originally brought in to move the company to agile development methods, and I spent the first year teaching them scrum and agile methods in general. Then as agile methods and that mindset took root in the company, we began expanding agile thinking into other areas of the business, particularly in operations.

As we leverage more agile methods, tools, and processes in our operations group, we’re getting to the point where I feel the business has a good foundation so we can actually practice what I call business agility. The company is getting poised to respond to different market conditions and to leverage our IT resources, services, and processes so we can react better.

We also have a good foundation for an innovation engine, and we have the ability to iterate rapidly on new product ideas. Like most companies, we’re trying to find new markets and new customers.

PwC: It sounds like you’ve explored the DevOps philosophy as well.
JE: I have a lot of experience with it.

PwC: What is Ancestry’s approach in that respect?
JE: We are looking at the public cloud, but the basic approach we’ve taken now is to develop a private cloud. When I came to the company, we had
the traditional data center. Now we’re smack in the middle of a heavy effort to virtualize our data center and to provide more infrastructure services, just like a private cloud would.

**PwC:** Which specific requirements were most pressing in your case?

**JE:** The biggest requirement was to reduce the time it takes to get something set up and out to the marketplace. Although development time is required for those things, the big long pull was that of our IT infrastructure. When working with physical machines, it would take as much as a couple of months to get new services provisioned and set up. So we had a long lead time, we didn’t have the flexibility to move resources around, and existing hardware was dedicated to current services.

**PwC:** Was storage also locked in?

**JE:** To some degree, although not as much because we had network-attached storage solutions that were a little more flexible. And our network was fairly flexible. The biggest problem was our hardware utilization. Whenever we needed to set up a new product, application, or service, we had to set up new machines, provision them, and all that goes along with it. That was our most difficult problem.

**PwC:** What were the key ingredients of getting the infrastructure to be agile?

**JE:** The number one challenge I dealt with initially was education and mindset within our operations group. Once, shortly after I was hired, I came in and said, “We want to deploy code at least once a day. We want to get to a place where we can deploy our code daily.” People basically fell out of their chairs. That just wasn’t heard of. To them, that idea was big red flags all over the place.

I wasn’t escorted out, but that idea of daily deployment was really frightening to them. What I meant was that we need to change our mindset—we can’t be a constraint on the business. I wasn’t even saying we have to roll every day, but that we need to be able to roll when the business wants to roll—and if that’s daily, then we need to do that.

Before, IT would say we could release only in particular windows—only with a certain frequency or every so often. If you needed to roll outside of these windows, then we’d have to go through a bunch of red tape to make that happen.

Challenge number two was to persuade operations that it’s possible to change by changing our processes, our tooling, and some of the technologies we use.

**PwC:** How did you reduce the dependencies?

**JE:** We had our architects tell development, “You need to be able to release independently from anyone else and if you can’t do that, you must change what you’re doing.” And we put in more stringent controls to ensure backward and forward compatibility.

Sometimes we still have coordinated code rolls, of course, but in the last couple of years we’ve made a lot of progress. Different components in the architecture are much more independent. Now we can roll smaller pieces more independently, and if one of those things goes wrong, then we immediately know, “that was the thing that broke,” and you roll it back or fix forward.

**PwC:** What happened after you got past those immediate challenges?

**JE:** Fairly quickly we were rolling code consistently on a two-week basis. Not every development group would release code every two weeks, but the operations group had a two-week release cadence. And their experience was that every two weeks was a nightmare. We would have problems. So when we told them we wanted to go to daily, they took their experience that they had every two weeks and thought, “I’m going to experience that every day.”

In their eyes, we had multiplied their pain by ten. And that’s why they were very resistant. But I pointed out that if you can increase the number of deployments and if every deployment were smaller, each deployment would go easier. The deployments they were doing every two weeks were huge—they had many moving pieces and interdependencies—and so invariably something was going to go wrong.
PwC: Is there some kind of road map we could talk about that takes a step-by-step approach to becoming more agile and that gets into what the open source community is doing a lot of?

JE: Honestly, when I started, I didn’t have a road map. In hindsight, I have a road map. I can articulate very well the steps we went through, but at the time, the first step was to move into agile. At first, the effort was principally development and it also involved our product management group. Then the effort worked its way up into the executive ranks, where this idea of delivering value more frequently and readily started surfacing other issues. Those issues led us back to the infrastructure.

When you tell development teams that every two weeks they’re going to produce working software that should be deliverable to the customer and they say “Great,” but they can’t release it or get it to the customer, the lack of delivery points to other problems that must be resolved. Principally, those pointed to our operations area.

As I mentioned, we talked to the architects and we started pushing back and asked, “How can we architecturally decouple components?” Then the next phase was the concept of continuous delivery into the organization.

At this point we’ve standardized on that whole notion of continuous delivery. Each group is basically allowed to release value to the customer as they see fit.

PwC: Did you set some limits on the way they could do this continuous release?

JE: Yes, but it was more indirectly enforced through tooling and automation. It’s not a hand process. In the new method, everything is done in an automated fashion. As long as they conform to the automation and the tooling, then they’re fine. They really can’t release code on their own; they basically push a button and the code goes out.

Ultimately, we put a lot of responsibility and ownership on that team to own their service. With the continuous delivery model, the whole idea is that anytime you check in code, that code must progress through a series of quality gates and testing. It must be deployed into preproduction environments and tested in preproduction environments. Then, if it makes it through all those gates, it’s deemed ready and the team can opt to push the button and release it into production.

We hold the teams responsible. If they release new code into production and there’s a problem, we’ll do a postmortem and say, “The code you rolled resulted in a production outage of some severity. What happened, and how can we help you correct that problem?”

PwC: Do you have any metrics on the number of code rolls you’ve done, and how many resulted in a problem?

JE: In the last two weeks—that’s 10 days—we’ve done 103 code rolls. Of those 103, two resulted in a problem in production, but neither of those affected customers. In one case, the problem was detected and corrected within five minutes, and in the other case it took about an hour.

The absence of serious production problems points to the fact that we’re rolling smaller increments of functionality, and so it’s a lot easier to make sure the quality level is high.

PwC: Do you have an example of how the frequent code rolls have had an impact?

JE: One of the principal case studies I have used was our 1940 census release in 2012. Things have matured even more since then. A lot of new features were released to the website to take advantage of the new census data in a different way. For example, we offered more advanced searching, including advanced image searching and image browsing. In that process, we released 16 new services in the infrastructure that supported that data release. During that campaign, we started to leverage this newer infrastructure and continuous delivery.
PwC: What sort of transition was it for your people, and to what degree did you find it necessary to bring in new talent who had experience with Chef or some of the other tools you’re using?
JE: My group is called engineering productivity, and that name came about because the focus of my group is to improve the productivity and the ability for the business to deliver value to the customer more rapidly.

My group was created from the ground up. I hired folks who have particular skill sets. We didn’t really want to disrupt the core, so the traditional operations kept going in the same vein. I hired a new team that has more specialization in automation. I looked for a skill set that combined a sysadmin-type person with a developer. In our operations group, we had more sysadmin-type folks or even network-type folks. But we really didn’t have too many developer folks. So I went after that combo skill set. From there, we could build up the automation that was necessary to build up the tooling.

“**The absence of serious production problems points to the fact that we’re rolling smaller increments of functionality, and so it’s a lot easier to make sure the quality level is high.**”
Blueprinting continuous delivery inside the enterprise

Lothar Schubert and Laurence Sweeney of CollabNet compare different enterprise maturities and ways to bring each type closer to a continuous delivery model.

Interview conducted by Bo Parker, Bud Mathaisel, Robert L. Scheier, and Alan Morrison

Lothar Schubert
Laurence Sweeney

Lothar Schubert is senior director of solution marketing and Laurence Sweeney is vice president of enterprise transformation at CollabNet, a development platform as a service (DPaaS) provider.

PwC: How do established large enterprises begin to move to a continuous delivery model? Where do they start, and how do they then integrate what they have developed using this new capability back into the core?

Lothar Schubert: We have been working closely with customers and with some of the analysts on this question, and the result was that we put together a blueprint for organizations to deploy enterprise cloud development. The blueprint considers existing practices that we see in organizations of different sizes across different industries, and it consists of five steps, or principles, as organizations mature in the development processes. (See Figure 1.)

The first step [embrace the cloud] is really about standardization as well as consolidating and centralizing assets associated with your development processes, not just about doing stuff somewhere in the cloud. It’s as simple as centralizing the uses and providing common access to shared resources in the organization—a common view on code, development processes, and development artifacts, as well as documents, wikis, and requirements.

The second step [implement community architecture] starts with mapping the business and enterprise architecture according to a taxonomy that allows different stakeholders to work together effectively. Particularly in globally distributed, large organizations, different stakeholders work on different projects. You need to have one common view and to structure your development portfolio and the different stakeholders according to the taxonomy.
PwC: Does that effort require an enterprise architecture?
Laurence Sweeney: My focus is working on these kinds of transformations, and my experience has been that when new tools such as Subversion and Git and some of the practices such as agile and DevOps come into traditional enterprises, they’re not usually brought in by the execs. They’re quite often brought in by the practitioners, and quite often it’s under the covers.

Certainly before 2009, no one had heard the word DevOps, and it was quite common for portfolio managers to focus very tightly on requirements and ensure value was being delivered. It was kind of OK that there was a black box in the middle. As long as the developers were happy and didn’t make too much noise, stuff more or less happened.

But in many cases, organizations have ended up with a very fractured development environment. It’s not unusual to walk into an enterprise and find that they have every tool under the sun, sometimes two of them. And now they want to start doing things like agile.

You start having real problems if the source code is in two or three dozen different repositories, if requirements management is all over the place, and if some teams are using some of these more modern tools and they’re completely disconnected from the corporate portfolio management. You can end up with a nightmare very quickly.

So is a new enterprise architecture necessary? The answer is yes and no. It depends on how you want to approach the problem. I see people approach the problem in three major ways.

In the first way, people close their eyes and hope the problem goes away. In other words, they work on something else, and the problem just doesn’t make it to the top of the stack because another part of the business is hemorrhaging. Let’s set that case aside for the purpose of this conversation and look just at the people who have decided they want to make a change for good reasons.

In the second way, an enterprise architect designs all this beautiful stuff up front. The architect will lay out the hierarchies and taxonomy, and that’s a really good approach if you have the bandwidth, the right people, and the greenfield to do it. You’re essentially building a completely new greenfield environment and then moving people to that greenfield.

Forge.mil is an example case for this kind of greenfield opportunity. Basically it was an “if you build it, they will come” situation. It was a cloud development initiative for the DOD [US Department of Defense], and they did the enterprise architecture work up front. They laid out what they wanted to do. They built it. They constructed the projects and communities. I think that’s a very effective way to work.

Laurence Sweeney: The third way is sometimes a bit more pragmatic and contrasts with the full enterprise architecture approach. It is probably best described as draining the swamp. You don’t do the up-front work and build the full greenfield environment because you don’t have time.

Instead, the first thing you need to do is to get projects onto a cloud development platform such as CollabNet. You just get all the stuff into one place, or centralize it. You need to get your
“You need to find out where all the source code, requirements, and projects are. After that, the problem space suddenly gets smaller for most enterprises.”

arms around the problem. You need to find out where all the source code, requirements, and projects are. After that, the problem space suddenly gets smaller for most enterprises.

If you approach the problem space from the enterprise architecture point of view in the draining the swamp scenario, you will worry about everything that’s out there and will likely get bogged down. But if you are pragmatic, initially stand up the platform very vanilla, and just move people onto that platform, people will immediately start voluntarily coming out of the woodwork and saying, “Hang on a second. This product or this project is retiring in 18 months,” for example. An ROI discussion can follow, and it provides the data to make investment decisions. You may discover some significant part of your problem space is not worth moving. There are a variety of good reasons.

Perhaps you can archive it. Perhaps it’s a project that never paid off. You can remove those projects from the list. Everything else gets moved onto the platform. That’s the first step, but it doesn’t mean you can’t have your enterprise architects working in the background.

Getting back to Step 2, once you have corralled everybody into a single standard platform, you know where all your IP is. Ideally for your timeline, you also are giving some thought to how you want to structure going forward.

**PwC: What happens then in Step 3 [codify development processes] of your blueprint?**

**Laurence Sweeney:** With startups and modern greenfield environments, codified development processes are pretty much going to be agile and using a lot of the thinking from the DevOps community.

But if your enterprise has been running software for 10 to 15 years or more, you must identify which of your processes are agile and which are waterfall processes. Then you must ensure the right projects are in the right process. When you do an ROI analysis, you’ll find that some projects are executing just fine as waterfalls. There’s no business advantage to moving them forward as an agile project. On the other hand, for lots of projects there will be a business advantage to moving forward, so you want to invest in codified development processes.

**Lothar Schubert:** One company we worked with, Deutsche Post, is a really good example case of the codify development processes step of the blueprint. The company outsourced many of its development efforts and had about 200 vendors that did various developments for them. Deutsche Post also had the old application portfolio in the enterprise architecture. The challenge for Deutsche Post was that it had no qualification of the development processes and it had different metrics for every vendor. Every vendor used its own methodology and code repositories. This situation was obviously a nightmare for management.

In that case, you assess the quality of the projects delivered. You assess which projects—and which vendors—are going better than others. The best ones inform your codified development process and your standard set of templates for process, the tools you can use, and the metrics by which to measure vendors.

**PwC: What about Step 4?**

**Laurence Sweeney:** That step is to orchestrate DevOps. These steps imply a serial process, but in the best cases, you’re working on these as related work streams. Perhaps think of them as swim lanes with interconnected concerns.

Orchestrating DevOps is really about making sure you have automated your continuous delivery and integration. You need to make sure those things are available, so you can start collaborating enterprise-wide. For this step, you need to have good role-based access controls and allow people to see as much as can be seen within the constraints of the separation of duties. In a collaborative version control platform, a trust and transparency is created, but that doesn’t imply you throw away your role-based access controls or give up on separation of duties.

---

**PwC: Git is a software instantiation of Linus Torvalds’ model of trust. Is that model consistent with what you’ve just been talking about?**

**Laurence Sweeney:** If you look at the basic model of trust that is in Git [a social network for version control developed by Linus Torvalds of Linux fame; see the article, “Making DevOps and continuous delivery a reality,” on page 26 for more information], it’s a very technocratic model and it’s one I actually like as a developer. It often is referred to as the benevolent dictator model. Sometimes he claims to be benevolent, and sometimes he claims to be less so.

But what is problematic is the concept of compliance. Git is a distributed version control system, and as a developer it gives me great flexibility in how I refactor my code, what my code base looks like, and ultimately where I push my code. From a compliance point of view, however, if you check in something in Subversion, it’s there forever. There is no obliterate command. The only way to get stuff out of Subversion is an offline admin process with dump and filter. With Git and the rebase command, you can make history look like and say whatever you want it to look like and say in your repository.

At CollabNet [which developed Subversion], we solved that problem with TeamForge History Protect. The solution allows developers to do what they need to and compliance teams to see exactly what’s been done. It also provides a handy “undo” button. That’s very important.

Subversion has 50 percent market share for software configuration management right now, and a lot of the people running on that are agile. I don’t think Git will replace Subversion completely. We’ve seen a 50/50 split between those folks who wanted to work in Subversion. Some teams preferred a single stable trunk approach, which is Subversion. I think we’ll need to have two hammers in the toolbox for some time. They serve slightly different purposes.

One of the smart things the Git community did was design Git to be a client for Subversion. You can use Git quite happily on your desktop, and from the corporate compliance point of view, you’re working in Subversion.

**PwC: Finally, how about the fifth step?**

**Laurence Sweeney:** The fifth step is leveraging the hybrid cloud, or using both private clouds and public clouds. The most famous hybrid cloud use case I’m aware of right now is Zynga. The company had made use of the Amazon public cloud, but found that somewhat limiting and developed a private zCloud.² There’s still the whole concept of being able to use the public cloud as elastic capacity when you need it. I believe one of the studios has done some other things with its rendering farms.

The ability to use that public capacity when desired is certainly very important. However, it’s really hard to manage that capability in an organized business fashion if you don’t have your act together on the other stuff. If you really don’t know where your IP is, for example, that automatically becomes problem number one and is an indicator of your organization’s level of maturity generally.

---

Ancestry.com campus in Provo, Utah
For decades, CIOs have applied methods and adopted tools that lead to stability. With good reason: When an IT deployment goes badly, businesses lose customers and risk long-term reputational damage. CIOs can lose their jobs. So business units had to live with a rate of business process innovation that was tethered to the plodding pace of IT’s waterfall software development life cycle (SDLC). These IT systems methodologies are rigorous, but slow and linear.

“The requirements change so quickly that if IT’s pace is in weeks and months, you end up not delivering anything, because the problem changed by the time you deployed your solution,” says Ken Venner, CIO of aerospace company SpaceX.

Speed versus stability is a paradox for IT. Pushing for both at the same time seems contradictory, but the IT organization must strive to achieve this goal if the enterprise is ultimately to become more antifragile—that is, less vulnerable to catastrophe. (See the article, “The evolution from lean and agile to antifragile,” on page 06 for a further explanation of antifragility and its business relevance.)

Is it possible for the CIO to accommodate a fast-changing business environment while ensuring stability in IT deliverables? It is, but it requires breaking with the safe yet rigid workflow principles of the past. Some companies will be more comfortable with this break than others.

“We have to develop software in other than a Taylorian mindset, where we handed off detailed specifications to a software team to deploy,” says Jez Humble, a principal at ThoughtWorks Studios, a software development consulting firm.

IT can transform itself by adopting agility without giving up stability, but CIOs must change the IT mindset about how to achieve stability. As explored in this article, the key to this change is to adopt a DevOps philosophy that emphasizes speed and efficiency without sacrificing adequate stability. Specifically, CIOs can achieve IT agility and stability by making change a frequently recurring process, which they can accomplish through the adoption of appropriate organic, evolutionary practices used by antifragile, web-scale companies.
The IT agility opportunity through DevOps

Historically, CIOs have trained their organizations and their users that volatility is a threat to stability. IT has responded by adopting Information Technology Infrastructure Library (ITIL) and IT service management (ITSM) standards and practices originally formulated by the British government for technology contractors, similar to the US Department of Defense’s creation of the waterfall methodology. These standard practices made sense at the time, when the world moved at a slower pace. Quality outcomes resulted only from establishing rigor and cross-validation beginning with specifications, followed by disciplined stages of design, development, testing, and release. The big downside: lead times were extended to accommodate this discipline, and that pace does not match today’s business dynamics.

CIOs could strive to balance the emphasis on stability inherent in ITSM and ITIL with a DevOps philosophy that emphasizes speed and efficiency. DevOps enables a closer working arrangement between developers and operations personnel; its best practitioners automate where possible and in the process they make developers responsible for ensuring their code functions well in an operational environment. (See Figure 1 for a summary of the main characteristics of the DevOps philosophy.)

Tools that further DevOps and continuous delivery goals help developers gain visibility into the operations environment, streamline their workflow, and automate a number of build, release, and deployment steps. (See the article, “Making DevOps and continuous delivery a reality,” on page 26 for more on related tools.) That visibility is consistent with better controls. (See Figure 2.)

Speed versus stability is a paradox for IT. Pushing for both at the same time seems contradictory, but the IT organization must strive to achieve this goal if the enterprise is ultimately to become more antifragile—that is, less vulnerable to catastrophe.
There is no insurmountable inconsistency between ITIL/ITSM and agile methodologies/DevOps. (See the figure on page 13.) Agile methods add flexibility to accommodate variability, improve quality, and speed up the cycle. By using the tools, processes, and techniques described in the business and technical articles of this issue of the Technology Forecast, enterprise IT can change into an organization that offers strategic assets to support innovation, bring new IT products to market more quickly, and respond to dynamic business opportunities—all characteristics of antifragile enterprises.

“For enterprise IT, the whole environment has changed,” Humble says. “Not just in the tool chain, but also in customer expectations of a high-touch experience.”

Agile development, continuous delivery, and DevOps all require organizations to break down silos within the business, including the systems analysts, the software developers, the testers, and the release and operations teams. New features become available as soon as they are ready, allowing instant interaction and iteration, something waterfall processes allow only at the end of the release cycle.

ITIL and ITSM help solve the stability problem. Agility, continuous delivery, and DevOps help solve the IT speed problem. Achieving stable IT operations is a long-standing goal for both, with some major differences:

- A fundamentally different theory drives stability: As originally envisioned, ITIL and ITSM controlled the process of change and ensured stability by reducing the frequency of change and triggering an IT process if change was introduced. DevOps sees change and evolution in a different light, as a way to boost the enterprise immune system—a key to antifragility. A DevOps and
DevOps. The level of total IT risk proportional to speed to execution in risk. The risk curve is not directly methods, they can actually reduce total IT feedback and response cycle in their IT. If organizations move toward a constant alignment—what PwC calls total IT risk.

The challenge for IT is that the traditional approach to IT risk itself introduces delays and is overwhelmingly biased toward managing only small coding errors rather than the risk that IT might fail to deliver business advantage. Traditional IT risk management should be expanded to include technical debt and business model alignment—what PwC calls total IT risk.

If organizations move toward a constant feedback and response cycle in their IT environments with the help of DevOps methods, they can actually reduce total IT risk. The risk curve is not directly proportional to speed to execution in DevOps. The level of total IT risk drops as the commitment and ability to embrace DevOps methods grows.

Enterprises should debunk the commonly held IT department wisdom when it comes to risk, and reassess risk that factors in both the strengths and weaknesses of cloud infrastructure and highly collaborative software development. Jez Humble described what he considers to be the true open source risk picture this way: “Open source actually provides a great way to mitigate some risks because you have systems, packages, libraries, and software where the source code is freely available. Many eyeballs have been on that code. You have a large community of people who have the knowledge to use those systems and support them over time.”

In essence, large-scale collaborative development on its own can change the risk picture entirely, and enterprises would do well to take a fresh look at how cloud-based IT collaboration and automation affects both the magnitude and source of risk.

The table below describes some other major risk categories and compares traditional IT risk management with the DevOps/continuous delivery approach.

<table>
<thead>
<tr>
<th>Type of risk</th>
<th>Size of impact on operations</th>
<th>Traditional IT risk approach</th>
<th>DevOps continuous delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small coding errors</td>
<td>Small</td>
<td>Controls through infrequent code changes, user testing</td>
<td>Controls through automated testing of full software life cycle</td>
</tr>
<tr>
<td>Major coding errors and/or poor software acceptance</td>
<td>Large</td>
<td>Controls through slow, exhaustive testing</td>
<td>Controls through automated, graduated introduction of new software to user base with iterative feedback and revision</td>
</tr>
<tr>
<td>Business model shift reduces business/IT alignment</td>
<td>Large</td>
<td>None</td>
<td>Introduces partial solutions and rapidly adjusts with frequent code releases</td>
</tr>
<tr>
<td>Technical debt caused by infrequent updates</td>
<td>Large</td>
<td>Requests large budget for upgrades</td>
<td>Addresses technical debt routinely within budget</td>
</tr>
</tbody>
</table>

Continuous delivery philosophy also views gestation periods leading to big bang deployments as extremely high risk. The longer the gestation, the more likely the business model won’t work, even if the code does. Business requirements change too quickly for a multiyear development plan to succeed. DevOps and continuous delivery advocate smaller, viable deliverables that attempt to address smaller parts of the requirement piece by piece. The operating principle behind DevOps is to make the system amenable to many changes and less likely to have a massive failure.

- **The rhythm shifts from an IT view of how IT changes can be introduced to a business view of changes necessary to compete:**
  This view has been a fundamental gap in alignment between IT and business. Business units become frustrated that their internal IT team is slow to respond, expensive, and not error free. They don’t want to run their own IT, but they conclude they have no choice. IT is frustrated that business units go off on their own, possibly introducing a heterogeneous mix of components and apps that won’t work over time. Startup companies have been the most demonstrable examples of DevOps, but even older businesses can adopt these capabilities and may be in danger if they don’t. (See the sidebar “How two companies move toward the new model.”)

- **The rhythm of change is possible only with high levels of automation:**
  IT must place a tremendous emphasis on design for automation—especially for testing, but also for configuration management, release management, infrastructure provisioning, and other delivery steps. Technical staff should devote quality time to devising and deploying the automation capabilities. Automation will make an important difference in continuous delivery and DevOps proficiency and productivity.
The waterfall approach is still de facto in several industries that have extreme business sensitivity to privacy, data security, independence, or process transparency reinforced by regulations. Financial services must follow strict regulatory compliance rules, reinforced in their public financial accounting (for example, FINRA). Healthcare, nuclear utilities, defense, and other industries also have such constraints, and not every CIO has the opportunity to help transform an entire industry from the outset. However, CIOs in any industry could apply continuous delivery at least somewhere in IT.

There might be an opportunity for agility to gain a foothold in a subsegment of the business, such as a new division, a spinoff, some internal IT (such as HR), or a new acquisition. CIOs can then bridge from that subsegment to more segments of the enterprise after proving the concept and knowing its limitations.

PwC advisory director Keith Katz describes an online equities broker-dealer that started its agility journey by establishing the pair-programming equivalent of co-creation of specifications and code. (See the sidebar “How two companies move toward the new model.”) It doesn’t require an entire industry or an entire company transformation to set the stage.

CIOs must put aside the notion that perfect software specifications are even possible with more laborious methods. Perfect specifications may be an illusion, because IT assumptions of what is needed may not match what the users want. With the agile development approach that DevOps and continuous delivery expand, IT does not design for one point in time for all time. Illusory precision from the waterfall method often didn’t work, either. There is no single deliverable, date, and budget. Continuous delivery is, by nature, iterative. Conversations with the customer will be ongoing, and deliverables will be multiple by design.

The CIO action plan for agility
Deploying code daily is a jarring contradiction to everything seasoned IT professionals believe about stability. Fundamental changes are needed in people, processes, and technology. But those fundamental changes do realize benefits, as underscored in a recent Rebel Labs survey of 600 IT professionals, which contrasted the average workweek of traditional IT operations teams with the DevOps equivalent. (See Figure 3.)

The survey research confirms that more time spent on automation setup and general testing pays off in the reduced need for support, infrastructure management, and communications time. PwC recommends that CIOs implement the fundamental changes necessary for a DevOps environment.

Step 1: Address the IT skills gap
Competence with new tools and methods must be resolved early. “Technical competence is a major constraint,” says Charles Oppenheimer, founder of Prizzm. Agility is a new area for which most organizations have renegade talent but not organized competence. Some of the talent may be within enterprise IT or in shadow IT groups. Enthusiasts who already have the passion and some experience can form the basis of the team. Organizations also may need to contract some talent from outside. A banking industry CIO interviewed for this article is building on the substantial open source expertise within the bank, while bringing in outside mobility experience. “We may need someone to jump-start us,” he says.

Start by setting up a group in enterprise IT entirely focused on agility. Assign a leader who embraces agility and who grasps the concepts, opportunities, tools, and methods. Ancestry.com, for example, brought an engineering mindset about productivity to IT.
Enterprises may need to restructure their pay and corporate culture to attract and retain the necessary technical talent. Work with HR to assess and address this reality. In addition, resources should be assigned—not “planned” in the traditional sense of SDLC.

Step 2: Establish an agile and DevOps blueprint

Agility can’t be entirely without direction, so CIO leadership is needed to provide clarity on design goals, priorities, and resource assignments. The CIO needs a blueprint to ensure that the overall direction is consistent with the enterprise needs and resources.

Engage the business unit CXOs in a survey to identify the top five business opportunities for agility. The best candidates are scenarios where the dialogue with customers frequently changes and where the IT organization will need to iterate rapidly for new product ideas, markets, and customers.

Some initiatives already may be underway, set up independently from enterprise IT, which can be coalesced into the official road map. The CIO should not view this scenario as an opportunity to commandeer those initiatives, but instead should embrace them as pilot projects that are included in the opportunity road map and that have a goal of shared organizational learning.

Focus on the two-way conversations with the customer, or the potential opportunities for having these conversations, and on conversations that have a dynamic nature that warrants agility. Include IT suppliers, if they can add value, and by all means engage with businesses partners, especially suppliers, distributors, retailers, or staff in marketing, sales, and advertising.

---

**Figure 3: Overall workweek for DevOps vs. traditional IT operations**

<table>
<thead>
<tr>
<th>Activity</th>
<th>DevOps oriented</th>
<th>Traditional IT ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploying changes (application development, configuration and infrastructure updates)</td>
<td>4.0</td>
<td>4.5</td>
</tr>
<tr>
<td>Automating repetitive tasks (writing scripts/using automation tools)</td>
<td>5.3</td>
<td>4.6</td>
</tr>
<tr>
<td>Infrastructure management (reviewing evolving infrastructure needs)</td>
<td>3.5</td>
<td>3.3</td>
</tr>
<tr>
<td>Infrastructure improvements (test processes and recovery plans)</td>
<td>4.8</td>
<td>3.6</td>
</tr>
<tr>
<td>Support (handling support for ops infrastructure)</td>
<td>2.4</td>
<td>3.8</td>
</tr>
<tr>
<td>Communication (meetings, e-mails, planning, etc.)</td>
<td>5.1</td>
<td>7.2</td>
</tr>
<tr>
<td>Firefighting (e.g., mission-critical failure recovery)</td>
<td>3.8</td>
<td>4.8</td>
</tr>
<tr>
<td>Self-improvement (training, reading, education)</td>
<td>3.1</td>
<td>2.7</td>
</tr>
<tr>
<td>Overhead (socializing, brainstorming, daydreaming, procrastinating)</td>
<td>2.8</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Source: Rebel Labs, 2013
Step 3: Invest in technical tools, software management approaches, and infrastructure

The CIO must determine what technology investments are required to get started. Early priorities include the following:

- Choose a software management, version control, and change management and programming approach (such as pair programming).
- Choose a social collaboration tool.
- Automate everything.
- Establish the rhythm of continuous delivery and DevOps.

An early step for the agile IT team is to determine how to manage each experiment in the road map. Change management helps ensure that the user community is informed of the change and the rationale. Release management ensures that all testing is completed satisfactorily and that new code will not conflict with something else in code, security systems, or infrastructure. Change management and release management will help ensure that design inconsistencies are caught before the code is promoted to production.

Despite the flexibility required for agility and for the mindset associated with experimentation, agility does not translate into chaos if CIOs provide proper oversight. Most importantly, agility does not mean the abolition of change management and release management disciplines. Agility-experienced IT groups advise that rigorous change management and release management disciplines should be in place. When press-worthy major outages occur in the DevOps world, they are traced to gaps in change or release management, and are not attributed to agile IT.

Case study

How two companies move toward the new model

Several companies provide examples of IT transformation to the new IT solution delivery model. At SpaceX, an aerospace company, CIO Ken Venner has taken advantage of the momentum from an inspirational CEO and a new business strategy for aerospace, although aerospace itself has high regulatory and self-imposed standards. And within the traditional real estate business, CIO Andy Kelk used DevOps to meet the challenge of enabling a new business model for iProperty Group.

SpaceX

Perfect outcomes and the desire for stability are embedded in the DNA of the aerospace industry. Every aerospace engineer learns at the start of his or her training that a perfect outcome is mandatory. Aircraft and space vehicles should never fail, and that mantra continues through all design, production, testing, and operational processes. IT takes its cue accordingly, and most aerospace IT takes a traditional approach.

Ken Venner, CIO of SpaceX, has an IT background with large companies and years of experience with traditional approaches to major systems deployment. When he joined SpaceX in 2011, he found the opportunity to adopt continuous development methods. The CEO, Elon Musk, wants to create a new approach for aerospace, driving the company and inspiring continuous development.

“Perhaps that is because Elon comes from more of a high-tech background than an aerospace background,” Venner says. “IT releases code at least once a week, and we’re trying to move toward a continuous build and integration environment that uses test-driven development as we’re rebuilding the application suite that runs the company.”

Given a recent history of success, Venner and SpaceX have demonstrated that IT can effectively leverage its new IT delivery model for the business as it evolves to a new model for space launches.

iProperty

Another example is iProperty, which operates in an industry that has a very long set of traditions. Recently, iProperty chose to launch an entirely new area of business—commercial real estate—in a few months.

The IT development group was accustomed to “a very traditional approach in which IT code was turned over to QA, which would find all the bugs and then pass it to operations, which would run it and not touch it again until there’s a change request,” says CIO Andy Kelk.

That approach would not work in the short time window required to get iProperty into the commercial business. It was a challenge to “get the developers in the mindset that they are responsible for their code from the moment they start writing until it’s in production,” Kelk says. The successful application of agility and DevOps capabilities allowed the business to succeed. The developers saw the tangible benefits early and could make quick adjustments to software as iProperty deployed and then adjusted its business model.

As a result, “We’re able to deploy changes at any time of day—even multiple times a day—with very minimal risk,” Kelk says. For example, he adds, IT quickly responded to a change from a subscription model, which had been the foundation of iProperty’s residential real estate business, to a different revenue model for commercial property.
Nationwide Insurance began its agility journey by establishing change management for systems and for the organization, so roles and processes would be clear as solutions were deployed. Nationwide’s change management was a companion to the stepwise evolution of DevOps. Four years ago, Nationwide adopted a monthly release management schedule. “Before that, each of 26 business units had their own release schedule, which was chaotic for users,” says Vijay Gopal, vice president and enterprise chief architect at Nationwide.

Releases weren’t routine; each was unique and involved its own dependencies. IT was confusing users, and the organization needed to invest significantly in training each time. Moving to a monthly release schedule was a first step in Nationwide’s evolution toward agility, but an important and symbolic one. It demonstrated the consensus of the 26 business units that they had to trade some autonomy for a better, long-term result.

The infrastructure must be agile ready at the outset. Esser of Ancestry.com says CIOs should start on the technical dimensions by “asking themselves if the IT infrastructure can be leveraged, if it is an asset, or if the business believes it’s a liability.” Given the readily available cloud infrastructure for hire, answering those questions may be one of the easier tasks to accomplish quickly. “We’ve probably seen the largest gains on the infrastructure side,” Esser says. “Previously, even replicating a given server’s configuration was very challenging, if not impossible, and obviously that added a lot of time into the equation.”

Much of the automation required for agility assumes a private, public, or hybrid cloud architecture, and enterprise IT must provision it quickly. One way is for IT to act as a service broker. Phil Berman, a director in PwC’s infrastructure and operations practice, suggests that instead of setting up your own cloud infrastructure, it may be more practical and time efficient to go to a cloud provider (such as Amazon or Google) and acquire cloud services on behalf of the business, rather than the business negotiating for services on its own. “IT still keeps a level of control and governance around all IT, but basically resells Amazon or Google services back to the internal users.”

Berman describes a situation in which an enterprise IT organization entered into a “white label arrangement” with Amazon and Google, and through them provides cloud-computing services to the internal organization. Enterprise IT chooses the preferred providers, consolidates the purchase for lower costs, maintains the service catalog and governance over security, and charges the users for the cloud services and their value add.

Step 4: Automate

Currently, no tool automates everything. But automating various processes will improve agility. For example, reviewable scripts and logs can be created through automation. Then, if something goes wrong, the automation associated with that error can be fixed.

System setup and configuration can be automated (using Chef or Puppet, for example), but testing the setup is crucial. Automate code generation and integration, but test that, too. Add scripts that combine the code generation and integration. Test that, and so forth.

The fast-paced world of continuous delivery requires extensive pretesting, but techniques such as limited release also work well. In the limited release technique, code is rolled out to a sample of the production environment. If it does not meet expectations, the code is rolled back. If it meets expectations and proves stable, the code can be promoted to the enterprise universe.

Despite the flexibility required for agility and for the mindset associated with experimentation, agility does not translate into chaos if CIOs provide proper oversight.
Venner calls this method test-driven development. Some organizations might encourage different teams to experiment with different tools, especially at the outset. But it is also important to avoid the usual tendency to do extensive research before choosing just one tool and one approach.

**Step 5: Remove barriers**

The CIO’s job is to enable others to succeed. Unlike the CIO’s role in major SDLC projects, the CIO of a DevOps environment will not manage (some might say micromanage) the details of process work, design, development, or production. Instead, Venner says, “It’s finding, understanding, and removing blockers for the development team, which means having conversations around what obstacles they have—either in terms of the business and the business interaction, or the technology or some process that’s stopping them from getting the jobs done.”

Often, as many as 5, 10, 15, or 20 development teams will be working at the same time. The CIO can’t manage the details or the fast pace of everything, yet he or she will be responsible for the outcome.

The CIO alone can identify and resolve constraints such as infrastructure limits, preexisting contracts (outsourcing commitments or software licenses), business process traditions, organizational culture and behaviors, or lack of training. The transformation from a more traditional IT approach to a more agile-centric approach, the core being continuous delivery, will involve cultural and organizational changes.

The CIO’s horizon must encompass suppliers, channels, and customers. IT teams need the freedom to achieve agility and variability for the customer-facing systems (systems of engagement) while not jeopardizing the core legacy systems. Sometimes that balance isn’t clear, so the CIO must intervene with flexible judgment—agility in an executive sense. The technical tools and techniques may be new, but the CIO’s proven skills as a leader will help the organization succeed even in the face of many challenges.
Step 6: Integrate and modernize legacy systems

Most enterprise IT organizations have legacy software and data systems in place, such as enterprise resource planning (ERP), customer relationship management (CRM), and other systems of record. These systems run well after years of investment and tuning, and they are the core operations of the company. Eventually, CIOs will need to determine how to integrate the agile systems with the core and perhaps modernize the core itself.

Customer-facing systems, the systems of engagement, are an example. These systems may source data from the legacy core or create data that must synchronize with the core. In mobile banking systems, for example, taking a photo of a check with a smartphone to make a bank deposit requires an interface with the banking transaction and clearing systems.

Many legacy systems have significant residual sums on the organization’s balance sheets, and years of depreciation and life remain. These systems are a form of sunk cost, requiring a cost-benefit rationale for replacement. For some industries, the core systems may be subject to regulatory oversight, and for public companies, they are subject to extensive audit reviews.

Not all IT is amenable to agile approaches at the outset. A plan for modernizing the legacy systems will require a situation-specific analysis to disassemble the core and replace modules in a logical pattern. The tools and techniques for the legacy systems are the same as for new systems. To follow the principles of DevOps, a plan for legacy modernization must avoid long delivery cycles and big bang approaches.

Nationwide’s legacy modernization, for example, tackles the noncritical parts of the business first, proving concepts and adapting the user base. These systems are on a cloud, virtualized environment and have a fully secured interface back to the core. This approach allows “gearing” between the agile stack and the core through edge databases and edge applications. Nationwide’s design philosophy for replacement is to make the systems more intuitive, helping to ease the transition for the user base.

For those industries where the core is not readily amenable to agility, the CIO might consider spinning off the legacy into its own environment. Then wrap it with Java and application programming interfaces (APIs) so it can interface with the emerging new environment, and eventually phase it out.

The specifics of the blueprint—tools and methodologies—will vary depending on the new agile applications and the specifics of the core. The great news is that the building blocks are available. In fact, the IT organization may already have experience with many building blocks individually, but not in a holistic, agile way. The CIO has the opportunity to master these valuable capabilities to help transform the enterprise. The CIO should demonstrate agile IT leadership and then graduate to the next level.

Beyond IT: The CIO as agility leader for the business

To become more antifragile, a business will need agility, but it might lack experience in how to become agile. Sometimes the business is stuck because it applies its traditional approaches to change. Or, sometimes, business leaders think they are stuck because they operate in a nonagile IT environment. Either way, IT can prove itself capable in agile IT and serve as a role model for the enterprise.
Customer-facing IT services are the most strategic and are a logical starting point to help transform business processes. Customers expect a rapidly evolving relationship with your business that recognizes their needs and preferences.

IT has the potential to accomplish this transformation; now it needs to demonstrate its capability. “As we leverage more agile methods, tools, and processes in our [IT] group, we’re getting to the point where I feel the business has a good foundation so we can actually practice what I call business agility,” says Ancestry.com’s Esser, who learned that business process reengineering disciplines helped tremendously in applying agility concepts to the business.

The goal is to add strategic value to a dynamic marketplace. The method of choice builds on agile development and cloud computing with DevOps-centric approaches as well as ongoing evolution and delivery. Business agility will increase as IT agility does.

IT has the tools, methods, and infrastructure to adopt agility now. The CIO must be the agile leader for IT and for the business, because no one else has the toolkits or talents for this task. CIOs need to address the speed of business versus the speed of enterprise IT. CIOs also must lead the modernization of old thinking about stable IT systems. Certainly, no one else can harness the capabilities of the core IT legacy software and data—the systems of record.

Don’t force your internal business partners to go outside the organization or to watch as their competitive landscape is threatened. CIOs have the keys to agility and the unique leadership skills to demonstrate their value to the enterprise, both in IT and in the business overall.
An aerospace industry CIO’s move toward DevOps and a test-driven development environment

Ken Venner of SpaceX explores agile development’s latest evolution.

Interview conducted by Alan Morrison and Bo Parker

Ken Venner
Ken Venner is CIO of SpaceX, a space transport company founded in 2002.

PwC: What differences are you seeing in how developer groups in various industries approach more frequent forms of delivery?
KV: Although there are some very unique requirements when launching cargo and people to space, I’d say aerospace in general is a little bit behind other industries. But SpaceX—just based on the fact that our founder, Elon Musk [also co-founder of PayPal and Tesla], comes from more of a high-tech background than an aerospace background—inspires a continuous development effort. My development team is aware of how the build process takes place and how test-driven development is evolving at companies like Facebook, LinkedIn, and Google. Not all of it is applicable when dealing with spaceflight, but our goal is to take best practices from multiple industries to create the best possible solutions for SpaceX. Right now we release at least once a week, sometimes midweek, and we’re trying to move toward a continuous build and integration environment that uses test-driven development as we’re rebuilding the application suite that runs the company.

PwC: Are you using this build environment for your core systems, for systems of engagement, or both?
KV: With our ERP [enterprise resource planning] platform and financials package—systems of record—we’re adopting a strategy of weekly release, desired zero downtime, and continuous build, where we deploy capabilities
when capabilities are defined to be ready. The ERP platform uses a custom build application internal to SpaceX. Once a week, we roll out a new release of the application. Longer term, we’re looking at creating a promotion process that supports zero downtime deployment as well as the ability to trial features with targeted user groups, which would allow us to roll out smaller features more frequently.

PwC: Why would you want to make such frequent revisions to your core manufacturing and financial applications? We tend to think of those more as systems of record, where requirements don’t change that quickly.

KV: This approach might not make a lot of sense for enterprises that are not trying to dramatically transform or attack a new market segment and need to get new features.

At SpaceX, we are redefining how our industry operates, and therefore we’re creating new and different business processes.

And I work for a boss who is always interested in increasing efficiency and extracting steps from the process. Our company has a philosophy that no matter how well you’re doing, you can always do it better. Therefore, we’re always looking for a way to improve the process—to either increase automation or reduce a non-value-added operational step. To do this, we need to enhance, modify, or change our processes and the supporting systems.

PwC: Do you have some very specific requirements that might make a difference in whether—or to what extent and how—you adopt open source, as compared to a consumer photo-sharing site, for example?

KV: Well, embracing open source for internal-only consumption doesn’t have the liabilities or limitations that it would if it were the foundation of the product you’re bringing to market. Enterprises should be more willing to look at open source for internal consumption as an option and a solution, because they don’t really have an IT issue if they’re using it only for inside consumption.

PwC: In the past, you’ve mentioned a talent shortage. Can you talk a little bit more about where you find your talent, how you manage it, and the nature of the supply of the capabilities?

KV: We find talent from a variety of different backgrounds and industries. Some of the talent I’m finding tends to be fresh out of college. They’ve generally been raised in this more agile development environment; they have a stronger orientation to open source tools and not as much focus on the waterfall-oriented development strategy.

We can compete with companies such as Google, LinkedIn, and Facebook for developers, because our goal is to go to Mars. We have an intriguing mission that attracts top-end talent. They’re aligned with the mission of the organization and want to do something interesting and exciting.

PwC: What’s your approach to management and governance at SpaceX?

KV: It’s finding, understanding, and removing blockers for the development team, which means having conversations around what obstacles they have—either in terms of the business and the business interaction, or the technology or some process that’s stopping them from getting the jobs done.
At a higher level, it’s ensuring that the direction of all the development teams aligns with some larger vision. So that involves pulling up the team and making sure that as they finish building features, the features are taking the company in a particular direction and achieving a goal. I’m interacting with my peers to make sure the teams deliver—within the time frame—the features that are designed to win the business. And then we’re making sure we have the appropriate resources, and we’re managing the expectations of the organization.

**PwC: Related to those goals, how would you describe SpaceX’s business model?**

**KV:** At SpaceX, we use smarter engineering to create highly reliable rockets and spacecraft at a reduced cost. We believe our approach will help reduce the cost of launch down so low that there’ll be new use case models for it, and the features we will need so we can manage the number of people attempting to do launches will shift over time.

In general, though, the predominant approach is, “Let’s get into this thing and just start working it. Every day, we’ll figure out what piece we’re developing, whether to play it up or make it visible or wire it up underneath. And we’ll figure out if we’re trending in the right direction or not.”

**PwC: What process have you been through personally that led you to understand the potential of this agile, DevOps-oriented approach?**

**KV:** Well, I’ve always liked this area and have tried as hard as possible to focus on this leaner, more aggressive aligned feature set versus the big cascading waterfall model where you’re talking in years to get something done.

The requirements change so quickly that if IT’s pace is in weeks and months, you end up not delivering anything, because the problem changed by the time you deployed your solution. The business moves quickly here, and so one key challenge is investing enough time to understand how the technology stack works, how to use it, and how to change a thought process.

Once you start getting your arms around agile development, it’s really not that hard to think of a better way to do things. You just need to let go of some of the old paradigms you had and embrace newer structures and newer ways of getting things done.

In the old days when you had one development, one test, and one production person, creating code that messed up the world was relatively difficult to do. Today in the world of virtualization the attitude is, “I’ll just spin up another VM [virtual machine] and get another environment going.”

**PwC: How has the advent of the cloud changed the way development gets done?**

**KV:** Virtualization lets you spin up a large number of instances of your application environment. Therefore, the ability to have multiple parallel scrums going at any one time is relatively cost-effective and relatively simple to do nowadays.

For bringing new capabilities to an existing application or changing a business process, agile development is the fastest way to keep all players aligned, focused, and working on the same thing without having tons of delay. Users generally are not trying to talk to you about a problem they have today that you say you’re going to solve in a month. They have the problem right now, they need a solution right now, and they want to know what you will do for them right now. By using this more aggressive strategy, you basically get people to spend less time iterating on all the possible alternatives. They’ll invest their energy in the direction they’re heading in.

“We find talent from a variety of different backgrounds and industries. Some of the talent I’m finding tends to be fresh out of college. They’ve generally been raised in this more agile development environment; they have a stronger orientation to open source tools.”
“Once you start getting your arms around agile development, it’s really not that hard to think of a better way to do things. You just need to let go of some of the old paradigms you had and embrace newer structures and newer ways of getting things done.”
To have a deeper conversation about this subject, please contact:

Tom DeGarmo
Global and US Advisory
Technology Consulting Leader
+ 1 (267) 330 2658
thomas.p.degarmo@us.pwc.com

Chris Curran
Chief Technologist
+ 1 (214) 754 5055
christopher.b.curran@us.pwc.com

Bo Parker
Managing Director
Center for Technology & Innovation
+ 1 (408) 817 5733
bo.parker@us.pwc.com

Comments or requests?
Please visit www.pwc.com/techforecast or send e-mail to techforecasteditors@us.pwc.com
**Subtext**

**DevOps**

DevOps is a working style designed to encourage closer collaboration between developers and operations people: Dev+Ops=DevOps. Historically those groups have been known to work at cross-purposes. DevOps collaboration seeks to reduce the friction between the two groups by addressing the root causes of the friction, making it possible to increase the volume and flow of production code and to reduce the alienation of the operations people who must guard the stability of the system.

**Antifragility**

Antifragility takes its name from a popular business book, *Antifragile: Things That Gain from Disorder*, by Nassim Nicholas Taleb. In the book, Taleb describes an organizational management theory that appeals to some leading DevOps thinkers. Taleb is also the author of an earlier and related book called *The Black Swan* and a former trader. While black swans are the rare catastrophes that could cripple an industry (such as what the financial services industry encountered with credit default swaps), antifragility is a rethinking of how organizations should operate to survive assuming the eventuality of black swans. Conceptually, antifragile organizations actually thrive in disrupted environments.

**Continuous deployment**

DevOps encourages extensive automation and workflow redesign so developers can release small bits of code frequently (in a continuous delivery cycle) and yet not disrupt the operational environment in doing so. The workflow includes buffers, compartmentalization, and extensive monitoring and testing—a very extensive and well-designed pipeline, but also a rich feedback loop.

**Agile development**

Agile development started before the cloud, but now much developer tool innovation and collaboration occur in the public cloud. Native cloud development is emerging as a separate and very powerful socially networked phenomenon. Agile principles are consistent with, but not sufficient for, native cloud development.