A Consumer Products Perspective

Tech Trends 2014

Inspiring Disruption

Deloitte University Press

Contents

Introduction

WELCOME to Deloitte's fifth annual *Technology Trends* report. Each year, we study the ever evolving technology landscape, focusing on disruptive trends that are transforming business, government, and society. Once again, we've selected 10 topics that have the opportunity to impact organizations across industries, geographies, and sizes over the next 18 to 24 months. The theme of this year's report is *Inspiring Disruption.*

In it, we discuss 10 trends that exemplify the unprecedented potential for emerging technologies to reshape how work gets done, how businesses grow, and how markets and industries evolve. These disruptive technologies challenge CIOs to anticipate their potential organizational impacts. And while today's demands are by no means trivial, the trends we describe offer CIOs the opportunity to shape tomorrow—to inspire others, to create value, and to transform "business as usual."

The list of trends is developed using an ongoing process of primary and secondary research that involves:

- Feedback from client executives on current and future priorities
- Perspectives from industry and academic luminaries
- Research by alliance partners, industry analysts, and competitor positioning
- Crowdsourced ideas and examples from our global network of practitioners

As in prior years, we've organized the trends into two categories. Disruptors are areas that can create sustainable positive disruption in IT capabilities, business operations, and sometimes even business models. Enablers are technologies in which many CIOs have already invested time and effort, but that warrant another look because of new developments, new capabilities, or new potential use cases. Each trend is presented with multiple examples of adoption to show the trend at work. This year, we've added a longer-form *Lesson from the front lines* to each chapter to offer a more detailed look at an early use case. Also, each chapter includes a personal point of view in the *My take* section.

Information technology continues to be dominated by five forces: analytics, mobile, social, cloud, and cyber. Their continuing impact is highlighted in chapters dedicated to wearables, cloud orchestration, social activation, and cognitive analytics. Cyber is a recurring thread throughout the report: more important than ever, but embedded into thinking about how to be secure, vigilant, and resilient in approaching disruptive technologies.

For the first time, we've added a section dedicated to exponential technologies, working with Singularity University to highlight five innovative technologies that may take longer than our standard 24-month time horizon for businesses to harness them—but whose eventual impact may be profound. Examples include artificial intelligence, robotics, and additive manufacturing (3-D printing). The research, experimentation, and invention behind these "exponentials" are the building blocks for many of our technology trends. Our goal is to provide a high-level introduction to each exponential—a snapshot of what it is, where it comes from, and where it's going.

From a Consumer Products lens, we provided industry sector specific perspective on majority of the topics including CIO as a venture capitalist (how to leverage brand categories perspective for portfolio planning), crowdsourcing (specific strategies including crowdfunding, flexible workforce and data analysis contests), wearables (discussing the Empowered Employee and the Persistently Connected Consumer) and digital engagement (Omnichannel Brand Engagement, Ubiquitous Sensors and other topics).

Each of the 2014 trends is relevant today. Each has significant momentum and potential to make a business impact. And each warrants timely consideration—even if the strategy is to wait and see. But whatever you do, don't be caught unaware—or unprepared. Use these forces to inspire, to transform. And to disrupt.

We welcome your comments, questions, and feedback. And a sincere "thank you" to the many executives and organizations that have helped provide input for Tech Trends 2014; your time and insights were invaluable. We look forward to your continued innovation, impact, and inspiration.

Al Langhals
Principal
Deloitte Consulting LLP

Matt Law
Principal
Deloitte Consulting LLP

Karl Rupilius
Principal
Deloitte Consulting LLP

Darwin Deano
Senior Manager
Deloitte Consulting LLP

Enablers

Technical debt reversal

Lowering the IT debt ceiling

Technical debt is a way to understand the cost of code quality and the impacts of architectural issues. For IT to help drive business innovation, managing technical debt is a necessity. Legacy systems can constrain growth because they may not scale; because they may not be extensible into new scenarios like mobile or analytics; or because underlying performance and reliability issues may put the business at risk. But it's not just legacy systems: New systems can incur technical debt even before they launch. Organizations should purposely reverse their debt to better support innovation and growth—and revamp their IT delivery models to minimize new debt creation.

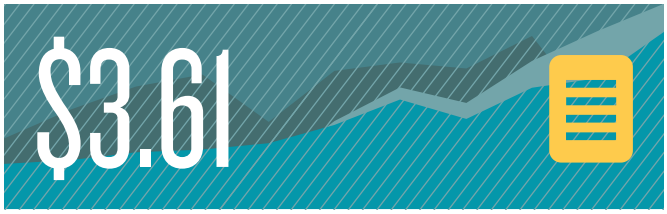
TECHNICAL debt is not a new term, but it's gaining renewed interest. Originally coined by Ward Cunningham in 1992, the phrase describes the "not quite right" code typically introduced with initial software releases because of an incomplete understanding of how the system should work.[1] Organizations that regularly repay technical debt by consolidating and revising software as their understanding grows will likely be better positioned to support investments in innovation. And like financial debt, organizations that don't "pay it back" can be left allocating the bulk of their budgets to interest (i.e., system maintenance), with little remaining to develop software that can support new opportunities.

Technical debt is often the result of programmers taking shortcuts or using unsophisticated techniques. It's typically misfeasance, not malfeasance. For example, a developer may copy and paste code blocks without thinking through the longer-term consequences. If the code ever needs to be updated, someone will have to remember to fix it in each instance.

But sometimes, technical debt is simply the result of dealing with complex requirements. To meet a project deadline, complicated proprietary code may be developed, even though simpler alternatives may have been available. With each such action, technical debt proliferates. This is like high-interest, short-term borrowing. If you don't pay off the debt promptly, compounding kicks in.

The impact of accumulated technical debt can be decreased efficiency, increased cost, and extended delays in the maintenance of existing systems. This can directly jeopardize operations, undermining the stability and reliability of the business over time. It also can stymie the ability to innovate and grow.

Articulating technical debt is the first step in paying off its balance. With new tools for scanning and assessing software assets, CIOs can now gauge the quality of their legacy footprint—and determine what it would cost to eliminate the inevitable debt. A recent study suggests that an average of $3.61 of technical debt exists per line of code, or an average of more than $1 million per system.[2] Gartner says that "current global IT debt is estimated to stand at $500 billion, with the



Technical debt per line of code within a typical application.[1]

The defect removal efficiency of most forms of testing.[2]

Estimated global annual expenditure on software debugging in 2012.[3]

Portion of total effort spent repairing architecturally complex defects, though they account for only 8% of all defects.[4]

potential to rise to $1 trillion by 2015."[3] While the idea of debt doubling in a year's time may seem astonishing, we're in the midst of unprecedented investments in disruptive technologies—often with deep hooks into core systems. The push for rapid innovation in unproven domains is also leading to compounding debt.

These estimates address only the literal definition of technical debt—how much it would cost to fix the exposed code quality issues. But there's also another dimension, which we call "architectural debt." Architectural debt refers to the opportunity costs associated with system outages or the inability to deliver new capabilities. In some cases, architecturally complex defects can absorb as much as 52 percent of the total effort spent repairing defects.[4] They can also derail new initiatives.

Technical debt is not limited to legacy systems; every new project has the potential to add to the backlog. With that in mind, you should incorporate the cost of technical debt into project management processes and portfolio reporting. This kind of transparency can not only raise awareness of quality among development teams, but can also provide a foundation for talking to the business about the hidden cost of IT delivery. By documenting your debt-decreasing efforts, you can account for those efforts—important progress that

Sources: [1] Alexandra Szynkarski, "Time to start estimating technical debt," October 29, 2012, http://www.ontechnicaldebt.com/blog/time-to-start-estimating-technical-debt, accessed December 27, 2013. [2] Namcook Analytics LLC, "Software defect origins and removal methods," July 21, 2013, http://namcookanalytics.com/software-defect-origins-and-removal-methods, accessed January 6, 2014. [3] University of Cambridge, "Financial content: Cambridge University study states software bugs cost economy $312 billion per year," https://www.jbs.cam.ac.uk/media/2013/financial-content-cambridge-university-study-states-software-bugs-cost-economy-312-billion-per-year/#.UryqUGRDsS4%20, accessed December 27, 2013. [4] B. Curtis, "Architecturally complex defects," December 19, 2012, http://it-cisq.org/architecturally-complex-defects, accessed December 27, 2013.

would likely not otherwise be visible (or appreciated).[5]

The ability to quantify technical debt can provide a common point of reference for the C-suite when you are deciding how to prioritize IT projects for an organization. Typically, technical debt should be paid down within the context of delivering against business priorities by incrementally refactoring existing solutions and using improved development processes to minimize new debt accumulation. Incorporating techniques described in our *Real-time DevOps* chapter[6] can help reduce waste generated when software is developed.

Some organizations may also need to spur projects that address especially messy issues such as bolstering performance, preventing production issues, or preparing for future strategic investments. The goal is a sustained, prioritized reduction of the balance sheet, where each project systematically improves on the baseline.

For most organizations, technical debt comes with the territory, an unavoidable outcome of decades of technology spend. The big question is: How will you manage the liability? Understanding, containing, and mitigating technical debt can be a platform, not only for a stronger IT foundation, but for a renewed level of trust and transparency with the business.

Consumer Products Perspective

The issue of technical debt sometimes does not get as much attention in the Consumer Products industry as in other industries like Retail or Transportation – the logic being that the prevalence of defined ERP packages in CP makes the problem far less acute. However, addressing technical debt is still a major challenge for many CP organizations. Companies often have a significant footprint of custom code potentially due to ERP customizations with ABAP or service-oriented integrations for enabling enterprise transformation logic (ETL) across the landscape.

As a result, the incurred technical debt—whether from poor software architecture and software development decisions or evolving needs—shackles the rate of change possible within the IT landscape reducing agility and increasing software development complexity. To reverse technical debt, there are a few time-tested practices that CP CIOs should consider:

- **Link technical debt reversal to portfolio design** – Keeping a separate portfolio bucket for either landscape modernization or agility enhancement can protect the sanctity of these investments – and helps safeguard that other priorities don't trump these needs.

- **Create visible architectural artifacts** – The pithy saying "out of sight, out of mind" is even more relevant to the world of CP, where technology innovations like analytics or big data are revolutionizing the use of IT. Hence, it is important to create simple architectural artifacts (like technical debt heat maps) that clearly indicate the current state of the landscape – and which can serve as visible reminders for the organization – as strategic planning fever takes effect before the start of the new fiscal year. Annual reassessments of these technical debt maps are important to gauge the progress of the organization in alleviating technical debt.

- **Sensitize the Architecture Review Process (ARP)** – Develop a set of simple metrics and diagnostics such as lines of code or the number of coding concerns flagged by static code analysis tools that the ARP can use to assess and suggest mitigations for as periodic architecture reviews and approvals for projects are conducted. Also, incorporating static code analysis tools within the SDLC process is important for CP organizations to consider.

Lessons from the front lines

Express delivery of quality

To keep up with the over 150 billion pieces of mail delivered each year,[7] the United States Postal Service (USPS) depends in large part on the quality and effectiveness of its IT systems. So when quality concerns became apparent during one of its IT modernization projects and the USPS was facing budget concerns, USPS leadership proactively took action to manage the organization's technical debt.

First, USPS used the SQALE[8] method for assessing the quality of its technical debt across four software dimensions: reliability, performance, security, and changeability. With a clearer picture of how much technical debt existed and where, USPS developed a roadmap to remediate the critical software issues and transition to long-term sustainment following CISQ[9] standards. For example, USPS instituted automated unit test scripts, minimum code coverage testing levels, and static analysis of the source code. These changes improved application quality and performance.

Going forward, USPS is also applying these same measurable standards to other projects by including them as standard oversight and acceptance criteria in their statements of work. And because incorrect project estimates can introduce technical debt, USPS is revamping its project estimation techniques by requiring the use of both parametric and bottom-up estimating techniques. With these changes, USPS is starting to see both improved quality and more accurately planned IT costs across its portfolio.

Cleaning up shop

DB Systel, a subsidiary of Deutsche Bahn, is one of Germany's leading information technology and communications providers, running approximately 500 high-availability business systems for its customers. In order to keep this complex environment—a mix of packaged and in-house–developed systems that range from mainframe to mobile—running efficiently while continuing to address the needs of its customers, DB Systel decided to embed processes and tools within its development and maintenance activities to actively address its technical debt.

DB Systel's software developers have employed new tools during development so they can detect and correct errors more efficiently. Using a software analysis and measurement platform from CAST, DB Systel has been able to uncover architectural hot spots and transactions in its core systems that carry significant structural risk. DB Systel is now better able to track the nonfunctional quality characteristics of its systems and precisely measure changes in architecture- and code-level technical debt within these applications to prioritize the areas with highest impact.

By implementing this strategy at the architecture level, DB Systel has seen a reduction in time spent on error detection and an increased focus on leading-practice development techniques. The company also noticed a rise in employees' intrinsic motivation as a result of using CAST.[10] With an effective technical debt management process in place, DB Systel is mitigating the possibility of software deterioration while also enriching application quality.

Countdown to zero technical debt

NASA's Mars Science Laboratory project was classified as a "flagship mission"—the agency's first in almost a decade. It was a $2.5 billion project to land a car-sized, roving science laboratory, Curiosity, on Mars. The rover launched in 2011 and landed on Mars on August 5, 2012, with the continuing objective of determining whether Mars ever contained the building blocks for life.

Building a roving science lab is an immense challenge. Curiosity is an order of magnitude larger than any rover that had previously landed on Mars: It weighs almost a ton, stands seven feet tall, contains a robotic arm that could easily pick up a person, and includes a laser that vaporizes rocks. Curiosity's software is essentially the brain of the rover—integrating its many hardware functions to provide mission-critical functionality such as the descent and landing sequence, autonomous driving, avionics, telecommunications, and surface sample handling.

The software initially developed for Curiosity was inherited from previous rover missions. The core architecture was developed in the 1990s on a shoestring budget. The Curiosity project put approximately four years of work into building on top of that architecture for NASA's most complex mission to date. As the launch date approached, NASA started to see that the project wasn't coming together: The software had bugs and inexplicably failed tests; there were issues with the hardware and the fabrication of key components.

The project faced a difficult question: Do we push on towards a 2009 launch or delay the mission? The unique aspect of launching a mission to Mars is that the opportunity only exists once every 26 months, when Earth and Mars align. If they delayed the launch two years, there was a risk that the project might be cancelled altogether.

The project team decided to postpone the mission and began an incredible regrouping effort. The software team had to quickly decide whether to fix the current software or to start over completely from scratch. Given the existing software's technical debt, it was unlikely they could determine the magnitude of the lurking issues, or confidently plan for new project milestones. The decision was made to tear down the foundation and rebuild using the old code as a reference.

The team started from the beginning: revisiting the requirements, software design, coding, and reviews, and testing and implementing standard processes. The team instituted what they called the "Power Ten," a set of 10 basic rules each developer followed. The team developed coding standards, implemented multiple automated code analyzers and testing tools, and established a cadence of releases—one every four months. They unit tested every line of code and instituted code reviews early in the development lifecycle. Two hundred code reviews produced 10,000 peer comments and 25,000 tool comments—each one reviewed and resolved.

The results were staggering: 3.5 million lines of code, over 1 million hand-written, across 150 different modules. But this time, the numerous bugs and unexplained failures were gone. The standards, though they required additional work, added stability and quality. And with the fresh start, the team were adamant that technical debt be minimized—building a new foundation for future missions.

Though NASA's approach required a remarkably difficult decision, the results were worth the effort. The world can now watch as Curiosity tells us more than we ever dreamed we might know about Mars. And the achievements of the mission led to the announcement of a new $1.5 billion mission to Mars in 2020.

Combating system complexity

Military Health System (MHS), a unit within the United States Department of Defense, provides billions of dollars' worth of health services to over 9 million beneficiaries.[11] Facing enterprise-wide budget cuts, MHS began looking for ways to provide the same level of care with reduced resources. With dozens of IT systems built over 20 years ago, including clinical systems, supply chain, and billing, MHS recognized that reducing its technical debt was one way the organization could reduce its IT budget and improve business efficiency.

MHS embarked on a transformation with portfolio rationalization at the forefront in an effort to streamline its investments. Using an application health grid that removed potential subjectivity, MHS measured the business value, technical maturity, and cost of each of its systems. Business value was determined by how many business processes are supported. The technical maturity analysis focused on four areas: external stability (an evaluation of third-party software, hardware, and associated vendors); internal stability (an architectural evaluation); system availability; and security. The rationalization effort helped MHS identify over a dozen systems with high levels of technical debt that could be decommissioned—saving the organization over $50 million in ongoing operating costs within the first phase of the transformation.

MHS continues to use data-driven analytics implemented through SEMOSS.[12] The transformation that began with portfolio rationalization has now moved into optimization and dynamic portfolio planning. Reviewing a system's technical composition in combination with functional capabilities allows MHS to protect itself against future technical debt and make informed decisions about its overall IT portfolio.

My take

Larry Quinlan, global chief information officer, Deloitte Touche Tohmatsu Limited

Technical debt doesn't just happen because of poor code quality or shoddy design. Often, it's the result of a series of good or necessary decisions made over time—actions individually justified by their immediate ROI or the needs of a project. There are many examples: skipping a software update or infrastructure upgrade because there wasn't a clear business benefit; building point-to-point interfaces into a small departmental app to get it into the business's hands more quickly; choosing a product you already own to build a prototype in order to avoid a drawn-out vendor selection and procurement process.

The path to technical debt can be paved with good intentions, but when combined, can lead you to quality and architectural issues.

But good intentions don't give you a pass to ignore technical debt. Leading IT organizations can, and should, actively manage and reverse technical debt. These organizations have a vision for robust platforms ready to fuel growth and use nimble business-aligned delivery models to innovate, fulfill unexpected business-driven requirements, and ultimately solve business problems.

There are two aspects that are important to technical debt management. The first is to know where you stand. Reversal starts with visibility—a baseline of lurking quality and architectural issues. Develop simple, compelling ways that describe the potential impact of the issues in order to foster understanding by those who determine IT spending. Make technical debt a metric that your IT organization is conscious of—not just in planning and portfolio management, but in how projects get delivered.

The second is with the actual management of technical debt. There are a couple of ways to approach it: a big bang approach that fixes everything at once (which almost never works) or a selective approach to systematically reduce the backlog. Consider what is needed in the next year or two to assist with achieving your strategic goals. This will allow you to identify the parts of your portfolio that should be upgraded to achieve those goals. When it comes to each of your platforms, don't be afraid to jettison certain parts.

At Deloitte, we deliberately separate our IT budget into core and business-driven investments so business users can choose investments driven by their priorities. A server upgrade rarely trumps a functional requirement when battling for fixed investment funds. That's why architecture, platform, and technical debt investments are part of our core investment bucket—with priorities set by the IT organization. My philosophy is: What's the point of having a CIO if I need a committee to approve every upgrade? By keeping the core investments separate from the business-driven investments, we are able to avoid the technical debt we might otherwise accrue.

Preventing technical debt requires a philosophy that addresses the known and expected requirements with an underlying, agile platform. CIOs need the courage to make the investments that reduce technical debt—and the knowledge and the team to know where and when to make those investments.

Where do you start?

Technical debt calculation can begin when you have clear visibility to the quality of code for legacy systems as well as projects on the horizon. Only with both sets of information can you make the trade-offs necessary to manage technical debt effectively. For companies eager to get ahead of the technical debt curve, here are some important steps:

- **Assess the status of code for all significant investments.** Calculate your technical debt. Know the size of the hole you're in—and whether or not it's getting deeper. Evaluate the importance of each system to understand whether the technical debt has to be addressed—and in what timeframe. Aim for surgical repairs when possible, but recognize that some aging systems may be beyond incremental fixes. Prevention is preferred, but early detection at least allows for a thoughtful response.

- **Find out how future investments are dependent on your legacy systems.** Is your architecture ready for new initiatives? Can it scale appropriately? How well are back-end complications understood and fed into planning efforts? Should you launch legacy modernization efforts now to get ahead of impending business demands?

- **Think through the availability of talent to support debt remediation.** For some aging systems, your resources may not be sufficient for cost-effective updating. Talent should be factored into your analysis. Think of it as a multiplier on top of the raw technical debt calculation—and use it to define priorities and timelines.

- **Hold developers accountable.** Consider rating and rewarding developers on the quality of their code. In some cases, fewer skilled developers may be better than volumes of mediocre resources whose work may require downstream reversal of debt. Regularly run code complexity reviews and technical debt assessments, sharing the results across the team. Not only can specific examples help the team improve, but trends can signal that a project is headed in the wrong direction or encountering unexpected complexity.

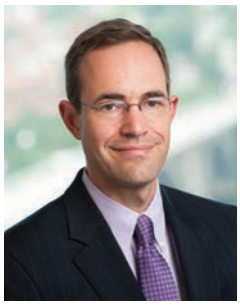
- **Spread the wealth (and the burden).** Communities are great ways to identify and address technical debt. Peer code reviews are leading practices for informal spot checks. Formal quality assessments by seasoned architects can find issues that would be undetectable with standard QA processes. Learn from open source communities, where quality is continuously refined by the extended pool of developers poring over each other's code.[13]

- **Determine your debt repayment philosophy.** Companies have different profiles when it comes to debt for the various parts of their asset pools. Debt is not inherently bad; it can fuel new investments and accelerate product launches. But left unchecked, it can be crippling. There's no single right answer for the appropriate amount of technical debt, but its accumulation should be a conscious, transparent decision.

Bottom line

When CIOs operate like venture capitalists,[14] technical debt is a big part of the financial picture. Without a clear view of the real cost of legacy systems, CIOs lack the information required to make effective decisions about new initiatives and investments. While it's important not to get obsessed with technical debt, it's also critical to understand and plan for it. Every new project automatically comes with technical debt as a cost of doing business. Reversing technical debt is a long-term investment, but if left unaddressed, it can bankrupt your ability to build for the future. Capers Jones, a long-term technical debt specialist, once said: "If you skimp on quality before you deliver software, you end up paying heavy interest downstream after the software is released for things you could have gotten rid of earlier, had you been more careful."[15] He was right.

Authors



Scott Buchholz, director, Deloitte Consulting LLP

Scott Buchholz is a technology leader with 20 years of experience in the areas of solution, enterprise, and data architecture; program management; and IT service management. He leads technology-enabled business transformations, from optimization efforts to full lifecycle implementations.

David Sisk, director, Deloitte Consulting LLP

Davis Sisk is a director in Deloitte Consulting LLP's US Technology practice. He has extensive experience in the architecture, design, development, and deployment of enterprise applications, focusing on the custom development area.

Endnotes

1. Ward Cunningham, "The WyCash Portfolio Management System," http://c2.com/doc/oopsla92.html, accessed December 31, 2013.
2. CAST, "Technical debt estimation," http://www.castsoftware.com/research-labs/technical-debt-estimation, accessed January 9, 2014.
3. Andy Kyte, "Measure and manage your IT debt," Gartner, Inc., August 9, 2010 (last reviewed June 19, 2013).
4. B. Curtis, "Architecturally complex defects," December 19, 2012, http://it-cisq.org/architecturally-complex-defects/, accessed January 9, 2014
5. David K. Williams, "The hidden debt that could be draining your company," *Forbes*, January 25, 2013, http://www.forbes.com/sites/davidkwilliams/2013/01/25/the-hidden-debt-that-could-be-draining-your-company/, accessed December 21, 2013.
6. Deloitte Consulting LLP, *Tech Trends 2014: Inspiring disruption,* 2014, chapter 10.
7. United States Postal Service (USPS), *Progress and performance: Annual report to Congress 2012*, http://about.usps.com/publications/annual-report-comprehensive-statement-2012/annual-report-comprehensive-statement-2012.pdf, accessed January 9, 2014.
8. Software Quality Assessment based on Lifecycle Expectations (SQALE) is a generic, language- and tool-independent method for assessing the quality of source code.
9. The Consortium for IT Software Quality (CISQ) is an IT industry leadership group dedicated to improving IT application quality.
10. "German national railroad operator ensures quality management with CAST," *IT Expert Magazine*, May/June 2013, pp. 38-39.
11. Military Health System (MHS), 2012 *MHS stakeholders' report*, http://www.health.mil/Libraries/Documents_Word_PDF_PPT_etc/2012_MHS_Stakeholders_Report-120207.pdf, accessed January 9, 2014.
12. SEMOSS, "Context aware analytics," http://semoss.org/, accessed January 9, 2014.
13. Simon Phipps, "Oracle's closed approach keeps Java at risk," *Infoworld*, April 26, 2013, http://www.infoworld.com/d/open-source-software/oracles-closed-approach-keeps-java-risk-217297, accessed December 21, 2013.
14. Deloitte Consulting LLP, *Tech Trends 2014: Inspiring disruption*, 2014, chapter 1.
15. Joe McKendrick, "Will software publishers ever shake off their 'technical debt'?," *ZDNet*, January 26, 2013, http://www.zdnet.com/will-software-publishers-ever-shake-off-their-technical-debt-7000010366/, accessed December 21, 2013.

Appendix

Authors

Bill Briggs
Chief technology officer
Director, Deloitte Consulting LLP
wbriggs@deloitte.com

Disruptors

CIO as venture capitalist
Tom Galizia, principal, Deloitte Consulting LLP
tgalizia@deloitte.com

Chris Garibaldi, principal, Deloitte Consulting LLP
cgaribaldi@deloitte.com

Cognitive analytics
Rajeev Ronanki, principal, Deloitte Consulting LLP
rronanki@deloitte.com

David Steier, director, Deloitte Consulting LLP
dsteier@deloitte.com

Industrialized crowdsourcing
Marcus Shingles, principal, Deloitte Consulting LLP
mshingles@deloitte.com

Jonathan Trichel, principal, Deloitte Consulting LLP
jtrichel@deloitte.com

Digital engagement
Christine Cutten, principal, Deloitte Consulting LLP
ccutten@deloitte.com

Barbara Venneman, principal, Deloitte Consulting LLP
bvenneman@deloitte.com

Wearables
Shehryar Khan, principal, Deloitte Consulting LLP
khans@deloitte.com

Evangeline Marzec, specialist master, Deloitte Consulting LLP
emarzec@deloitte.com

Enablers

Technical debt reversal
Scott Buchholz, director, Deloitte Consulting LLP
sbuchholz@deloitte.com

David Sisk, director, Deloitte Consulting LLP
dasisk@deloitte.com

Social activation
Dave Hanley, principal, Deloitte Consulting LLP
dhanley@deloitte.com

Alicia Hatch, principal, Deloitte Consulting LLP
ahatch@deloitte.com

Cloud orchestration
Andy Main, principal, Deloitte Consulting LLP
amain@deloitte.com

John Peto, principal, Deloitte Consulting LLP
jpeto@deloitte.com

In-memory revolution
Mike Brown, principal, Deloitte Consulting LLP
mikbrown@deloitte.com

Doug Krauss, specialist leader, Deloitte Consulting LLP
dkrauss@deloitte.com

Real-time DevOps
Ayan Chatterjee, principal, Deloitte Consulting LLP
aychatterjee@deloitte.com

Alejandro Danylyszyn, principal, Deloitte Consulting LLP
adanylyszyn@deloitte.com

Exponentials
Bill Briggs, Chief technology officer
Director, Deloitte Consulting LLP
wbriggs@deloitte.com
With contributions from Singularity University faculty and leadership and Marcus Shingles, principal, Deloitte Consulting LLP.

Contributors

Aaron Sotelo, Abdi Goodzari, Adarsh Gosu, Amy Bergstrom, Andrew Luedke, Angel Vaccaro, Ann Perrin, Antonio Caroprese, Chad Clay, Chrissy Weaver, Dan LaCross, Dan McManus, Daniel Ledger, Daryl Jackson, Dennis Startsev, Derik Quinn, Ed Panzarella, Elizabeth Rielly, George Collins, Gina Marchlowska, Irfan Saif, Jarrod Phipps, Jeff Powrie, John Daab, John Keith, John Stefanchik, John Sprouse, Jon Wiesner, Jostin Darlington, Junko Kaji, Kavin Shelat, Keith Zalaznik, Kevin Weier, Kumar Chebrolu, Lisa Iliff, Maria Gutierrez, Martin Hougaard, Matt Lennert, Missy Hyatt, Navin Advani, Nicole Leung, Oliver Page, Paul Krein, Paul Roma, Paul Toler, Prabhu Kapaleeswaran, Rajeswari Chandrasekaran, Ram Venkateswaran, Rithu Thomas, Robert Kasegrande, Sandy Ono, Steven Bailey, Steven Shepley, Tara Newton, Travis Budisalovich, Trey McAdams, Troy Bishop, Vladimir Baranek, Yu Zhu

Consumer Products Contributors
Darwin Deano, Richard Kupcunas, Matt Law, Russell McLean, Mukul Nagle, Oliver Page, Khelan Patel, Jarrod Phipps, Nitin Rao, Karl Rupilius, Shomic Saha

Research

Leads: Tom Carroll, Chris Chang, Tore Dyvik, Justin Franks, Thomas Gleason, Rui He, Thomas Henry, Karthik Kumar, Nicole Leung, Simy Matharu, Abhishek Mishra, Jose Munoz, Paridhi Nadarajan, Akshai Prakash, Fatema Samiwala, Jeremy Young

Team Members: Jacob Artz, Anwar Ayub, Rachel Belzer, Simeon Bochev, Kevin Bojarski, Mark Brindisi, Alex Carlon, Felix Cheng, Judy Chiu, Eugene Chou, Ian Clasbey, Kyle Collins, Kevin Craig, Brian Cusick, Philip Davis, Michael Davis, Jefferson DeLisio, Zach Epstein, Inez Foong, Marjorie Galban, Leksi Gawor, Rachana Gogate, Calvin Hawkes, Taylor Hedberg, Dan Heinitsh, Dan Henebery, Seimi Huang, Sam Jamison, Simon Jo, Solomon Kassa, Rebecca Kim, Ryo Kondo, Adrian Kosciak, Ashish Kumar, Varun Kumar, Corey Lian, Alyssa Long, Pulkit Maheshwari, Ryan Malone, Tyler Martin, David Melnick, Akhil Modi, Alice Ndikumana, Kashaka Nedd, Brittany Neisewander, Ryan Pallathra, Aaron Patton, Lee Reed, Talal Rojas, Tammy Ross, Jaclyn Saito, Hugh Shepherd, Will Shepherdson, Andrea Shome, Kylene Smart, Sam Soneja, Gayathri Sreekanth, Xenia Strunnikova, Lindsey Tsuya, Peter Van, Jordan Weyenberg, Jenny Zheng

Special thanks

Mariahna Moore—for being the heart, soul, and "buck" of this year's report—where every detail started and stopped, big or small. Your tireless leadership, spirit, and drive are truly inspirational and a singular reason we hit every ambition without compromising seemingly impossible deadlines.

Cyndi Switzer, Stuart Fano, Jill Gramolini, Kelly Ganis, and **Heidi Boyer**—the veteran dream team that makes Technology Trends a reality. Your passion, creativity, and vision continue to take the report to new heights. And your dedication, energy, and commitment never cease to amaze.

Dana Kublin, Mark Stern, and **Elizabeth Rocheleau**—for the tremendous impact made in your first year Tech Trending—from the phenomenal infographics to coordinating our volunteer army to jumping into the content fray.

Finally, a special thanks to **Mark White**, the founder of our Technology Trends report series and an invaluable contributor, mentor, and friend. Thanks for all of your continued support as we build on your legacy.

Recent Deloitte thought leadership

The Deloitte CIO Survey 2013

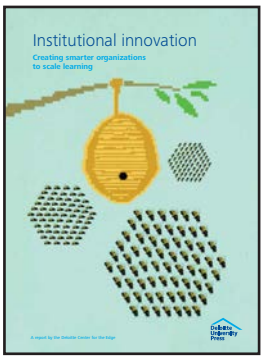
www.deloitte.co.uk/ciosurvey

2014 Technology Media & Telecommunications Predictions

www.deloitte.com/predictions2014

From Exponential Technologies to Exponential Innovation

http://dupress.com/articles/
from-exponential-technologies-to-exponential-innovation/



Institutional Innovation: Creating smarter organizations to scale learning

http://dupress.com/articles/institutional-innovation/

Stay connected with technology trends:

Subscribe to receive technology-related communications
www.deloitte.com/us/CIOSubscribe

Subscribe to the Dbriefs webcast series for technology executives
www.deloitte.com/us/techdbriefs

Other trends reports:

Analytics Trends 2014
www.deloitte.com/us/analyticstrends

Global Human Capital Trends 2014 (Coming in February)
www.deloitte.com/us/HRSubscribe

Business Trends 2014: Navigating the next wave of globalization (Coming in March)
www.deloitte.com/us/SOperspectives

Follow @DU_Press

Sign up for Deloitte University Press updates at www.dupress.com.

Learn more

Follow @DeloitteOnTech

www.deloitte.com/us/techtrends2014

About Deloitte University Press

Deloitte University Press publishes original articles, reports and periodicals that provide insights for businesses, the public sector and NGOs. Our goal is to draw upon research and experience from throughout our professional services organization, and that of coauthors in academia and business, to advance the conversation on a broad spectrum of topics of interest to executives and government leaders.

Deloitte University Press is an imprint of Deloitte Development LLC.