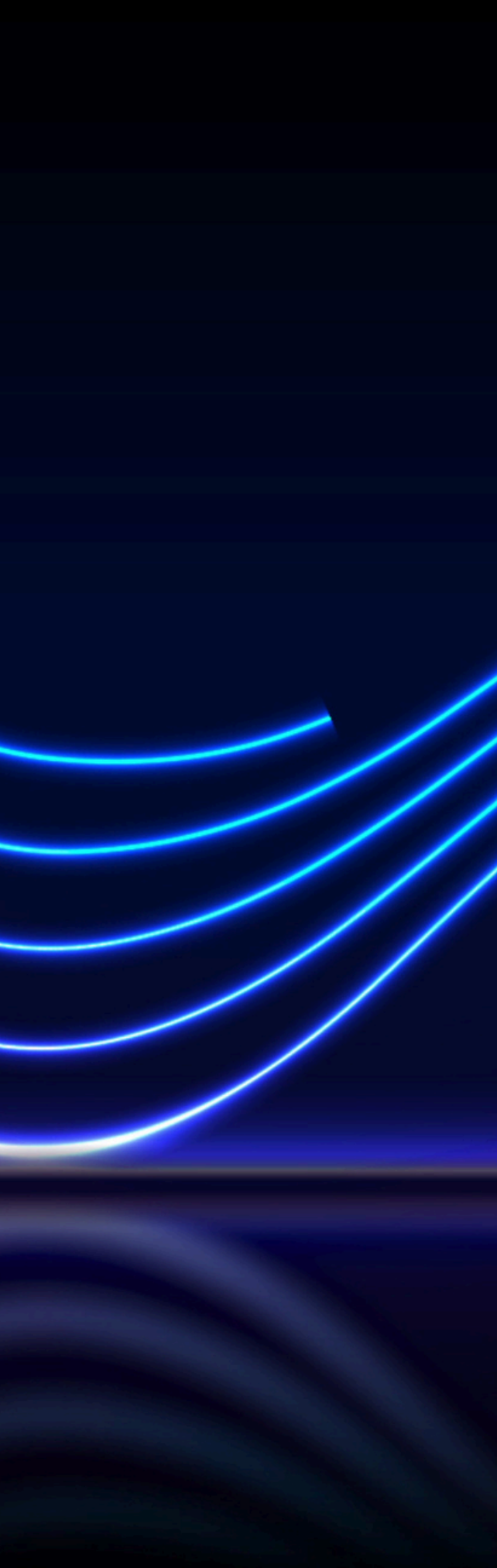


WHITEPAPER

Software Engineering: A Practical Guide to Modernization

There are several reasons that software engineering processes break down or don't optimally perform.





Software engineering is critical for innovation, which spurs business value such as improved customer satisfaction, greater employee productivity, higher operational efficiency, lower operational costs, and additional revenues from new products or services.

However, sometimes enterprises don't evolve their engineering processes to keep up with modern software development. This can cause significant problems — from technical debt to margin erosion and revenue losses.

That said, you don't necessarily have to rip and replace your entire software engineering practice. There are course corrections that will allow your enterprise to modernize and streamline.

Delve into this content to discover:

- The challenges holding back software engineering teams from being able to accelerate innovation
- The potential effects of inaction
- Three solution areas where organizations can make an impact
- How the right IT partner can help



What's Holding Back Software Innovation?

There are several reasons that software engineering processes break down or don't optimally perform. Here are three areas where practices tend to bottleneck or cause issues for the enterprise:

1 - Ad Hoc Software Development

This scenario may seem familiar: Software teams are given the application and business requirements, then design and code the software. They test it to a certain degree, and then deploy it. In other words, testing happens once, and any issues that arise after deployment are dealt with as they occur.

The problem with this ad-hoc approach is the significant potential for any one or all of the following: partial implementations, misconfigurations, and unstable and unscalable software. Testing should be embedded throughout the development life cycle, at every phase.

2 - Falling Short in Quality Assurance (QA)

Even those companies that test code throughout each development phase sometimes don't follow through with comprehensive QA testing.

It should run in parallel with development and the code should be verified by third parties, such as the International Standard Organization (ISO) quality standards.

Any shortfalls in the QA process can result in security vulnerabilities, bugs, or defects in the code. If these issues make it through deployment, the enterprise runs the risk of these holes being exploited by cyber-attackers or the software running at less than optimal performance.

3 - Not Taking Full Advantage of Tools

Companies increasingly using DevOps methodology to gain faster time to market, improved security, and greater code quality. There are multiple tools within the DevOps toolbox, however, many companies only use a handful of them or don't fully understand which tools are best for their purposes.

For example, they might adopt continuous integration tools like Jenkins or GitLab, as well as security tools for scanning and vulnerability assessment.



However, they might not use collaboration tools like Slack or Teams, which enable rapid communication among teams. Although your software engineers may not need every DevOps tool, it's important to understand all the possibilities and benefits. Avoiding the use of some tools — whether for cost or cultural reasons — can cause inefficiencies and software vulnerabilities. Why would enterprises not update their software engineering practices or take full advantage of toolsets? They may think it's too costly to invest in software tools, or that quick fixes or bolted-on solutions are sufficient ways to deal with software issues after deployment.

Sometimes workforce issues are at play. Companies might not have the right skillsets or expertise and are doing the best they can with what they have, yet still fall short. In other cases, there is cultural resistance to change. Even though DevOps initially came to market in 2007 or 2008, it took years for large enterprises like Amazon to embrace the methodology. Smaller companies may still not know enough about DevOps to adopt it.

Another issue is simply not realizing there's a software engineering problem. Processes may be working "well enough." It may take a misconfiguration or vulnerability for teams to realize their practices are not streamlined.



The Effects of Inaction

No matter the reason for process breakdowns or legacy approaches to software engineering, they all lead to a much more significant problem: technical debt.

This occurs when the costs to fix an issue continue rising and then start creating even bigger problems. Tech debt slows down the enterprise because it hinders the software development cycle, threatening the pace of innovation and business growth. In addition, companies are subject to other multiple risks:

- Increasing costs to implement new product features
- Customer frustration from sluggish app performance The potential for security vulnerabilities
- The inability to take advantage of cloud efficiencies and speed
- A loss in productivity for IT teams that must maintain older systems, as well as employees using these solutions

Solutions and Best Practices:

Although every business is unique in terms of its requirements, resources, and capabilities, there are strategies and technologies to help overcome challenges without having to reinvent the software engineering wheel. Here are three recommendations:

1 - Assessments

In this case, a trusted partner would assess your software engineering processes — for example, within product development, quality assurance, and DevOps. The provider would review existing software documentation, talk with architects and development leads, perform code reviews, and run code scans using a static code analysis tool.

The benefits are numerous. Assessments objectively uncover areas for improvement and categorize issues to fix according to your business's needs and risk appetite. You also gain the ability to create a technical path or roadmap to innovation.

Potential use cases for assessments:

Potential use cases for assessments:

- Pinpoint where a breakdown is occurring. Senior leadership may see the results of legacy systems and processes but are not able to identify the root cause. They suspect software quality is an issue due to product defects, poor app performance, user problems with the software, or unstable infrastructure.
- Unusually high development costs. An assessment can determine if overruns or excessive costs are due, for example, to inefficient processes or overstaffed or unskilled teams. Inability to add features. If the engineering team is taking a long time to add new features to an existing software product, an assessment can reveal where the problem lies.


An assessment typically takes two or three weeks and would include findings and recommendations. The right partner should prioritize the recommendations and provide estimates as to the cost and duration to fix.

2 - Automation

While many enterprises have incorporated automation into software engineering processes, there are opportunities for greater adoption — and benefits. For example, by reducing the need for humans to perform repetitive or manual programming tasks, the organization can accelerate development and reduce the potential for errors. Automation also frees up IT staff to think and develop more strategically, and more rapidly scale up new projects.

Potential use cases for automation:

- Quality assurance testing. There's a significant case to be made for automating regression testing to ensure any code updates or changes didn't introduce bugs that can affect the software's functionality.
- Robotic process automation (RPA). Similar to how it sounds, RPA is software robotics that uses automation to perform simple, repetitive tasks and business processes. It mimics mouse clicks and keyboard taps to automate user actions — saving valuable time for IT teams.



Ultimately, different forms of automation can accelerate the software delivery cycle and optimize processes.

3 - Artificial Intelligence/Machine Learning (AI/ML)

While there may be hype around AI/ML, these technologies are not part of a fad. There's real business value with the right planning and execution.

For example, AI/ML solutions can rapidly deliver insights or summarize data that would otherwise take humans inordinate amounts of time to obtain. These technologies also improve communication and collaboration among users who must shape AI/ML queries and models to ensure the right business outcomes.

Potential use cases for AI/ML:

- Code generation. Some large language models can write code snippets or even entire applications — however, it's critical that humans review the model's work.
- Bug detection. Rather than manually sifting through volumes of code, an ML model can rapidly do so to identify defects.
- Summarizing text. AI/ML solutions can ingest huge volumes of documents – such as reports and presentations — read them at speed, then summarize the highlights.

Adoption rates of AI/ML technologies grow each year. And your competitors are likely already testing them. But first, understand the use cases that fit your organization, and then be sure your teams know how to use these tools.

The Value of An IT Partner

RKON has helped clients through each of the scenarios mentioned above and many more situations. Sometimes issues come up during the due diligence processes of carve-out and merger transactions.

However, in many cases, our customers come to realize they cannot modernize or keep pace with business growth with the software engineering practices they have. They often encounter incidents when they try to put software into production, yet are unable to pinpoint the problem.

Our teams have deep expertise and experience across the software engineering spectrum. For example, we can help optimize the product development life cycle, improve your quality assurance program, and implement or streamline your DevOps operations.

Especially given the ever-more sophisticated cyberthreat landscape and competitive business market, it's becoming imperative to optimize software practices to not only keep pace but also gain the flexibility to more easily manage whatever issues arise.

No matter whether your business is grappling with the ongoing IT skills shortage, struggling to modernize development processes, or needing to better understand your software engineering capabilities, RKON is well-positioned to help.

ABOUT RKON

Founded in 1998 in Chicago, RKON has grown to become one of the nation's leading IT advisory practices. Our comprehensive understanding of execution strategies, technology, business processes, operations analytics, risk and compliance, and planning and integration supports hundreds of organizations.

Recognizing that no two companies have the same IT challenges, RKON takes a truly customized approach. We serve as trusted advisors to our customers, providing strategic guidance, technical resources, and honest assessments to address competitive challenges and meet long-term goals.

[CONTACT US](#)