4 Myths About DevOps in a Regulated Industry

18 March 2020 | **Cyber Security, Healthcare, Software Engineering** | **Rastislav Novotny**
Reading time: 7 minutes

I was recently talking to a partner working in IT in the finance industry. And he told me, *"It's not worth it to implement Agile and DevOps practices in a regulated industry."*

I was caught surprised by this perspective. Mostly because I've experienced not only successful implementation of DevOps in a highly regulated industry, but also having seen processes being enforced by security regulations. Therefore, I would like to break some myths about the idea of implementing DevOps in a regulated industry today.

Myth #1: A regulation comes into effect on a specific date. So its related project has a fixed time, fixed budget and fixed scope as well.

Regulations on businesses are mostly in place to protect the interest of customers and ensure the quality of a service or a product. To achieve that, regulations usually enforce a process to ensure quality. However, they do not dictate how the process must be implemented.

For example, the General Data Protection Regulation (EU GDPR) or the equivalent Personal Data Protection Act 2012 (PDPA) in Singapore says that the client has the right to ask a company for information about what data does the company holds about the client and how are they processed.

However, there are multiple options to implement this process. It can be a fully automated application that collects data from several CRMs and other systems, and exports these data to the client. Or just a simple email response provided by the legal or compliance team may be sufficient. Therefore, we see that the scope of a suitable solution is usually flexible.

A flexible solution means the project timeline is usually malleable too. While regulations in the finance industry may impose fixed constraints on financial transactions, different solutions lead to different costs per transaction.

This way, it may be possible to implement a simple solution within a short time and move on towards a more complex solution that helps lower cost per transaction in the future. This can too, be achieved gradually in multiple iterations in agile product development.

Myth #2: It's simply not possible to implement DevOps in a regulated industry.

For some time in the past, I've worked in the healthcare industry. The healthcare industry is

highly regulated to ensure proper care for a patient. Therefore, the minimal time to release a new version of a product was about 6 months, and usually even much longer.

A few years back, however, the Food and Drug Administration (FDA) in the United States released the cybersecurity regulation that any security vulnerability identified in a medical care product must be fixed no later than 60 days after the identification of the vulnerability.

This seemed like an impossible task in an industry where any fixes and release of a new product version could take at least 6 months. The solution was simple: shorten the release time to less than 60 days, which we achieved by adopting DevOps practices and automating as many tasks in the release process as possible:

- Automated approvals instead of manual signoffs.
- Automated testing instead of manual testing.
- Automated infrastructure provisioning and validation instead of manual.
- Automated generating of release/testing/traceability documentation.
  The first point is especially important, but also often the most difficult to implement. And because this usually means a change in organization and responsibilities of roles.

The main message here is that implementing DevOps in a regulated industry is not only possible but is even required by the introduction of new regulations. And while the strict cybersecurity regulation is in the healthcare industry now, very similar regulations may be adopted by other industries too. For example, there are already discussions about the enforcement of security vulnerability fixes in future updates of GDPR.

Myth #3: It's not possible to ensure the security of a system if there is a new version every 2 weeks.

The process for traditional software development includes several steps: specification, implementation, testing and validation, security validation, and other types of validation.

These steps are usually done separately by different departments or teams. The advantage of this process is a strong focus on specific aspects by experts with a high level of knowledge and experience. For example, a security expert with several years of security validation can provide a highly reliable report of the security status of a software system.

Unfortunately, this approach does not scale well. Especially when the security validation of a single software version can take up to several weeks or even months.

This problem can be solved by the "Shift Left" principle. The principle moves the task to earlier stages of the process. In terms of security validation, the principle says to include

security validation in the CI/CD pipelines and ideally include software security already in the system specification and design phases.

This may include multiple tasks:

- Define product backlog items with specific security goals.
- Static code analysis for best practices (e.g. not using obsolete cryptography algorithms).
- Scanning for dependencies with security vulnerabilities.
- Automated security testing.
- Validation of best practices for infrastructure configuration.
  With this, the task of a security expert is to enable teams to develop secure software. He/she should help teams to include security into pipelines and then to regularly validate and improve the pipelines. This way, the work of a security expert can scale much better.

Myth #4: Software developers are not responsible for security.

Going a few years back, most enterprise applications were deployed to application servers or complex runtime environments (e.g. J2EE or .NET Framework). The environment provides the infrastructure that applications usually require, including configuration, logging, exception handling, state management, throttling and more.

The advantage of this approach was that application infrastructure security could be managed by operations. For example, when a vulnerability was found in the authentication module of the application server, then the application server got patched and all applications on the server were secured without any involvement of the development team.

On the other hand, the disadvantage of this approach was that components were usually tightly coupled to runtime or application technology. This made testing of application components very complicated or almost impossible because the components couldn't run outside of the application server or in a very specific environment.

More agile frameworks were developed some time back (e.g. Java Spring or .NET Core). These frameworks provide inversion of control and dependency injection features to develop highly composable and extensible components. This way, it becomes much easier to implement unit-testing of such components.

In addition, these frameworks provide services for effective application bootstrap with minimal dependencies. Applications can then be deployed as self-contained packages with almost no dependencies on system-wide or shared frameworks. For example, applications can be deployed as Docker containers.

Unfortunately, this makes it impossible for operations to patch known vulnerabilities. Security patching must, therefore, be done as part of software development, and thus naturally falls into the responsibility of the software development team.

How can Agile and DevOps practices work in a regulated industry?

As the importance of cybersecurity grows more crucial in the digital age, I believe we are seeing that these new cybersecurity regulations will not impede, but rather embrace the implementation of Agile and DevOps practices in ensuring that providers are able to provide fast responses to security vulnerabilities.

This can be achieved by approaches like the “Shift Left” principle to move isolated security validation to security integration in the software development processes and CI/CD pipelines as well as leveraging enterprise-ready frameworks like Java Spring or .NET Core that are designed with this consideration.

* *The opinions, beliefs, and viewpoints expressed in this article are those of the author and do not necessarily represent the official policy or position of Zuhlke or its clients.*