OPTUM CASE STUDY

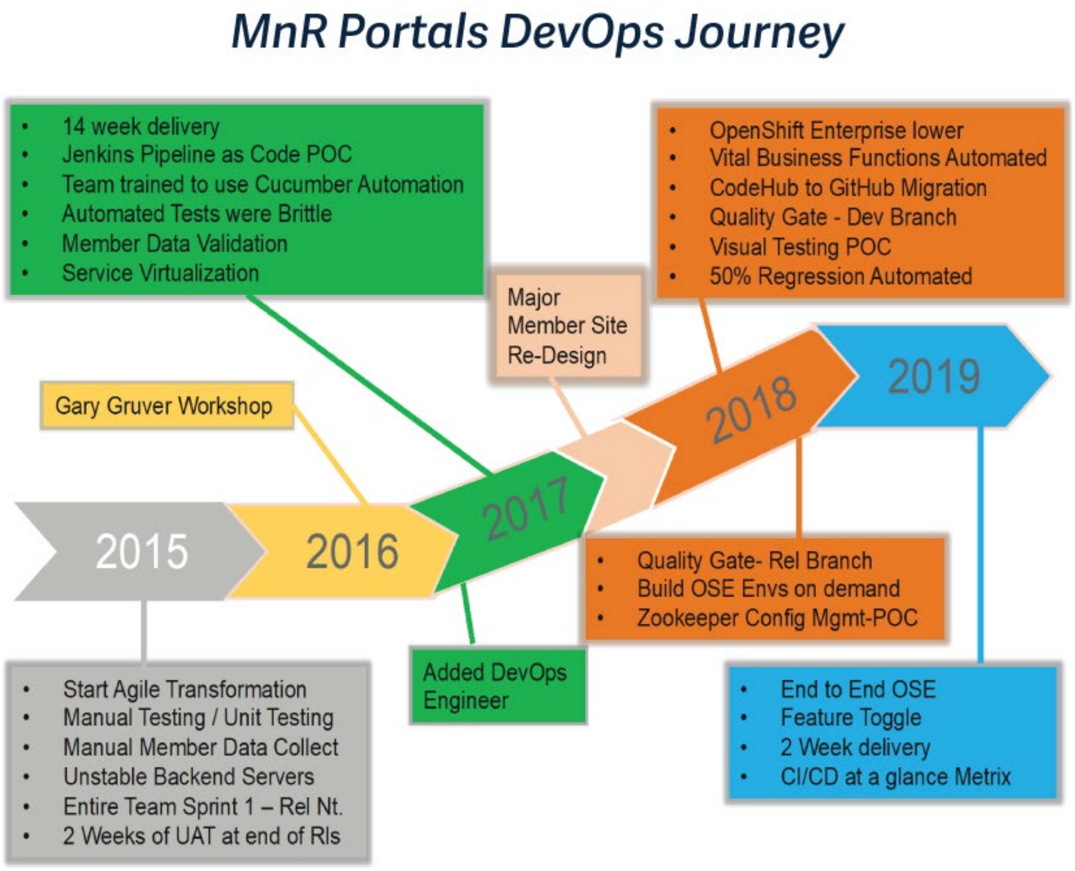
A good example of systematically creating a stable, quality signal is Ted Youel's journey at Optum Technology (Youel, Gruver, and Keyes 2018). Ted's team spent a few years going through and addressing all the stability issues in their deployment pipeline. When I first met Ted's team, they had been doing Agile development and test automation for a while. Each of the Agile teams were responsible for delivering new features and test automation every iteration. Their product development process would have five, two-week development sprints. Then they would have a four-week hardening phase where 100% of the team would move from doing feature development to inspecting quality and fixing all the defects. They had created this new test automation capability, but we found that they were only running it at the end of each iteration when they were signing off stories. Then during the hardening phases, they would run the Agile teams feature test automation again, along with automated and manual regression.

They were not running any of the tests on a daily basis. Like so many software organizations, they had the cultural norm of building up large amounts of inventory and then inspecting in quality during the hardening phase. One of the first steps we took was to focus on figuring out how to build more frequently and use the test automation to build in quality. Ted's team worked in a much broader system, so they did all the final testing in an integrated test environment. They realized that when they were trying to run their automated testing frequently, the endpoints from different teams were down or not working as expected. For them to get a stable, quality signal, they needed to be able to mock out that instability in the broader system using service virtualization so they could find code issues on their side. They needed to be able to deploy and test on a more frequent basis, which required investments in automation.

It took a while to implement the changes because, like most organizations, everyone was already fully booked delivering new features. But over time, Ted's team found the capacity to make changes and started seeing improvements. Where it used to take 100% of their team to inspect in quality and fix the defects for the four-week hardening phase, in 2017 they found they only needed half the team. They we were able to move the other half of the team to the next release and get started earlier.

They continued this journey of systematically improving issues that were impacting the stability of their quality signal over time, as shown in Figure 1.

Figure 1: MnR Portals DevOps Journey



MnR Portals DevOps Journey

2015

- Start Agile Transformation
- Manual Testing / Unit Testing
- Manual Member Data Collect
- Unstable Backend Servers
- Entire Team Sprint 1 – Rel Nt.
- 2 Weeks of UAT at end of Rls

2016

Gary Gruver Workshop

2017

- 14 week delivery
- Jenkins Pipeline as Code POC
- Team trained to use Cucumber Automation
- Automated Tests were Brittle
- Member Data Validation
- Service Virtualization

Added DevOps Engineer

Major Member Site Re-Design

2018

- OpenShift Enterprise lower
- Vital Business Functions Automated
- CodeHub to GitHub Migration
- Quality Gate - Dev Branch
- Visual Testing POC
- 50% Regression Automated

- Quality Gate- Rel Branch
- Build OSE Envs on demand
- Zookeeper Config Mgmt-POC

2019

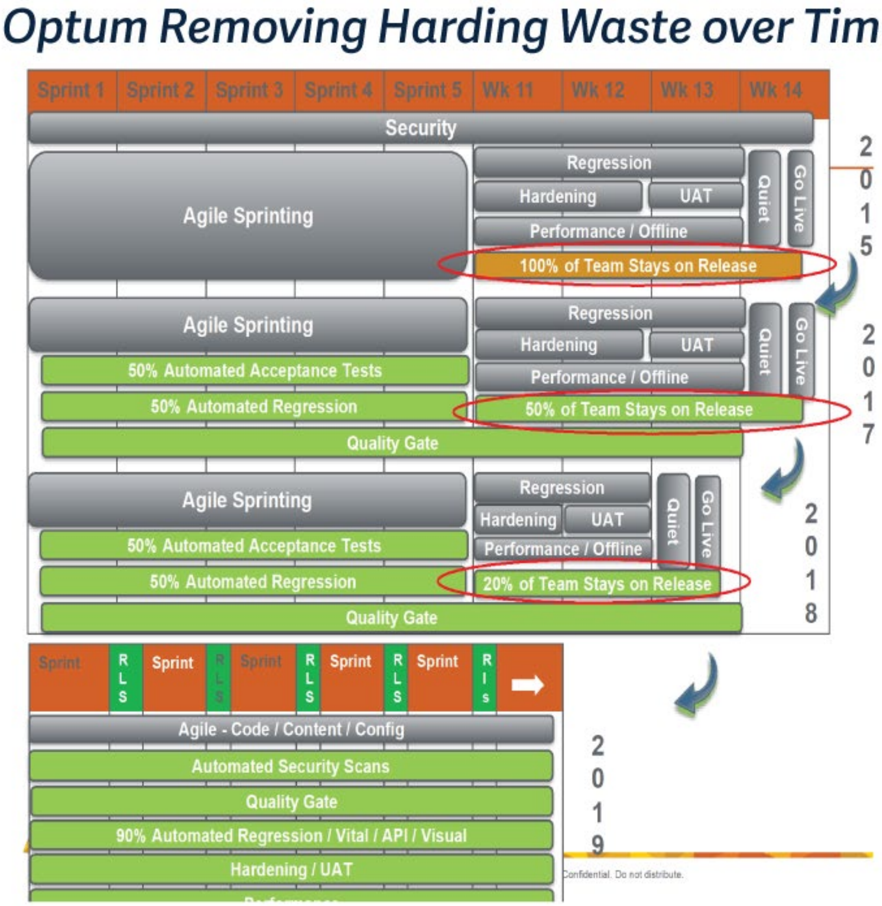
- End to End OSE
- Feature Toggle
- 2 Week delivery
- CI/CD at a glance Metrix

By 2018, they had a stable, quality signal they could use for gating code. It required automating most of their deployment pipeline and using service virtualization to isolate their team from the instability in the rest of the system. It also required reworking most of their automated tests because they realized they had a fair amount of stability problems with their current designs. They were using XPath instead of element IDs. They hadn't written their tests from an object-oriented standpoint, so they were having a hard time changing the tests fast enough to keep with changes in the product.   What they found is that the tests weren't stable enough to be used as a quality signal.

This is typical of most organizations. They have automated tests somewhere in their system, but when they start trying to use them to build in quality, they find they are not stable or maintainable. Most organizations have to rework their test framework and create automated test design patterns that are stable and maintainable before they have a stable, quality signal. Therefore, when I work with organizations, I always recommend they start by evaluating their test automation for stability and using it as a quality gate before they write any more tests that will probably end up needing to be reworked.

As Ted went through this journey of improving stability and building in quality, he found it dramatically reduced the amount of waste associated with the hardening phase. Over time, they were able to significantly reduce the amount of time and resources required for triage and fixing defects, as shown in Figure 2.

Figure 2: Optum removing harding waste over time



This is a really good example of what it takes to drive stability and the business benefits it can provide. Your journey will be different as your organization is unique, and it is going to vary for different deployment pipelines within your organization. The steps in the journey should be based on the requirements of individual applications and sources of instability. There isn’t one path that every organization should follow to reach a level of maturity to ensure success. Like in manufacturing, it is more important to apply the principles and then adjust the practices to address the specific application. The principle of quality is taking a systematic approach to creating a stable signal and then using that signal to build in quality. There’s a lot that we can learn from Lean Six Sigma if we’re not changing both the product and the process at the same time. Start by finding a set of automated tests that can be used to drive stability across the deployment pipeline. Then take a systematic approach to improving stability across environments, deployments, and tests. Once you create that stability, you can start using that fundamentally new capability to change your approach for how you do development. You can start to build in quality versus inspecting it in the large batch of inventory in the end game.

As Goldratt showed with the MRP system, instead of just automating tests to reduce the cost of testing that are run during the hardening phase, we can use this new capability running automation every day to build in quality. This is how Black & Decker used their MRP implementation to achieve lower inventory and better availability than anyone else. They didn’t just improve MRP by automating the manual process. They used the new capability to change how the entire organization worked by adjusting plans multiple times a week when they ran the MRP process on a more frequent basis.