# Vectoring Your Product to Success With CI

A necessary component of an agile team is to go beyond establishing a comprehensive continuous integration process.

Bill Drew

Wed, 06/19/2019 - 11:58

Photo Credit: shapecharge/iStock

One IT impediment for an organization looking to achieve an Agile posture and deliver cadence that supports its objectives includes the lack of a comprehensive continuous integration process.

Your CI process and environment need to support your development teams as they continually develop the functionality for your end user. But effectively getting to the product that’s best for your users takes more than just a comprehensive CI.

## The Product of Our CI

So you've done the hard work and you've spec'd out the requirements for your CI and the supporting environment. You went that extra yard and ensured that the implementation supplied the functionality and operational capabilities that meet those requirements. You've been diligently testing and validating the CI implementation. It's provisioned via IaC and scales against load. Unit, integration and even some level of performance testing can be performed. Static code analysis will drive quality and code metrics can be gathered and reported to ensure quality is trending upward.

Before you know it, your development teams are on board and producing their first builds. The code is being built, compiled, tested, packaged and integrated. The teams are proceeding toward their deliverables based on their interpretation of the requirements and acceptance criteria. At the end of the sprint, we'll have a look at what they've produced and see how close they've come to what the product owners had specified and envisioned.

## Why Gamble?

But why wait until the end of the sprint (or even the release if you really want to live dangerously) to see if development is on track? What about validating application and the workflow along with testing the functionality as soon as possible? Or as soon as a specific piece of some functionality, something that you're particularly interested in or was responsible for is available?

With your well-functioning CI process, there's no reason you can't. We just need one more thing: a feedback loop.

So what exactly do we need to do to get that feedback loop?

Well, first we need to establish an environment where the CI product can be deployed for validation and testing. This new environment should be in addition to any integration environment used by the CI process. If developers are adhering to the practice of regular commits to the code base, then this integration environment will be updated and changing with each commit and prove far too dynamic for POs and testers to properly interrogate and validate functionality. I refer to this new environment as the Inspect and Adapt (I&A) environment, as it will support inspection and feedback from users and development teams to adapt accordingly.

## Triggering Reviews

The decision to deploy an RC should be done by POs when they determine that a release candidate is functionality complete enough or some functionality has recently been added that they want to review and provide feedback on to the development team.

But POs will need information in order to even trigger that deployment.

First, there has to be notifications sent out by the CI process upon the successful completion of a build. This is something that should already be in place with your CI, and if not, it’s easy to implement.

Secondly, there must be a comprehensive and continually updated document containing the list of features and bug fixes that are included with each build, commonly referred to as release notes. These release notes should be created automatically with the CI process by extracting the relevant commits and comments from the VCS.

The effectiveness of this process and the quality of the release notes relies on developers posting not only quality comments but also relating them to a story and/or task identifier. Together the identifiers and comments will allow POs to better understand what’s in a RC and determine if it’s something that they’d like to interrogate and validate.

If it is an RC of interest, it’s tagged and deployed to the I&A environment and available to POs and testers. During a review, POs need to record any issues and enter them in the project’s issue tracking system. These issues need to be categorized and easily identified as “RC.x.x.x feedback” and traceable to a specific RC. It’s this careful review and detailed feedback on the current functionality that provides the crucial input into reviews with the development teams.

## Fine Tuning

After a review of the application state and recording their assessments, POs and development teams should meet to review and discuss. It’s here that POs can provide the critical and timely feedback on where they see the product going and how it may differ from their vision. It also allows the development teams to share with the POs why some functionality and features came out they way they did and offer possible alternatives.

These reviews should not be seen as a witch hunt nor as missteps by either party, but rather an invaluable opportunity to fine tune and vector the product and guide it to the end state that will best serve the end user.

The outcome of these meetings should produce new tasks and directives to the development teams, which should be reflected in their backlog of work and eventually make it into upcoming release candidates for review.

This feedback loop continues until the POs and testers are satisfied that the functionality meets the expectations for the release. We can then cut a release branch and start migrating our release candidate down the deployment pipeline to production.

## Steer Early and Often

Communication is challenging – especially when attempting to convey complex ideas and concepts. You should expect there will be misinterpretations and misunderstandings resulting in gaps in implementations and expectations.

But as any engineer will tell you, the earlier these are identified and remedied, the better off you'll be in terms of both costs and the end product.

When we marry DevOps processes and practices with Agile methodologies to produce an effective and continual feedback loop, we have the opportunity to identify these shortcomings and misinterpretations early and often. It also gives us the opportunity to apply fine-grained adjustments to the development effort and in the process put ourselves in a far better position to deliver the product that we, and our end users expect.

View printer friendly version
agile advocate
DevOps
modernization
Standard