YOUR SERVICE PROVIDER FOR

APPLICATION PERFORMANCE

www.triscon-it.com

Success Story with ITERGO

How Dynatrace empowers performance engineering teams to test a scale

Approaching Performance Engineering Afresh

Most of us are used to waiting until the very end of the software-development process to evaluate the performance of new applications. Many reasons are given for doing things this way, including the following:

- **Many feel that it does not make sense to determine the performance metrics of a system until the entire system is available and capable of running the new software.**
- **It is assumed that performance- and scalability-related issues can be identified in only a fully configured load-testing environment.**
- **It can be difficult to scale and apply the performance metrics of one environment to another, so we wait until our large, production-related server environments are ready before we begin performance analysis.**

These all make sense, but you can also see that there's a problem with this approach. By waiting, we risk delays in the entire process, and missed delivery goals.

Figure 3.1 shows a typical burn-down chart used in agile development. During development, the team works on user stories that could be fit into the iterations planned for the next release. Too often we see performance-related testing pushed to the end of the last development sprint. This approach is great if you need to push a lot of new features into a software product, but it jeopardizes the planned release date and can wreak havoc with quality standards.

Waiting until the end to focus on performance usually unveils problems that are too big to fix in the time planned for the testing phase. This either leads to missed goals as feature or quality cuts have to be accepted, or it leads to missed deadlines.

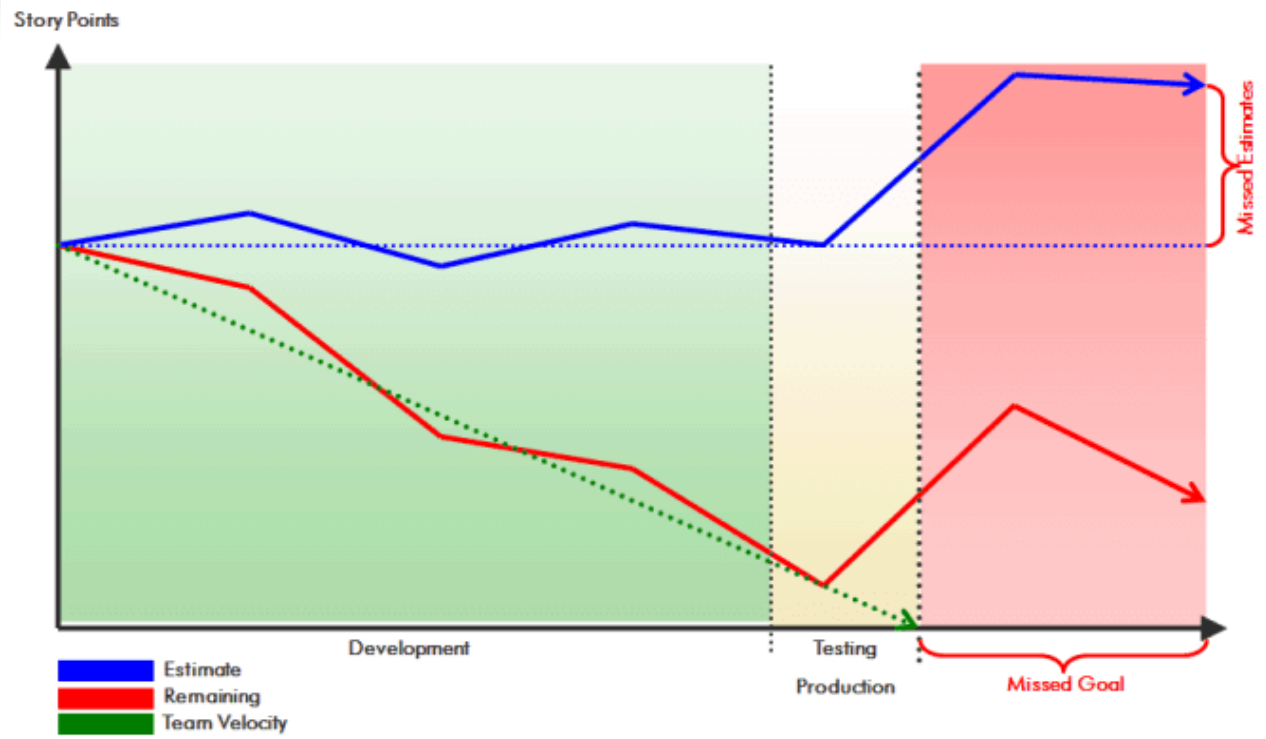

Figure 3.1: Pushing performance testing to the end of the release cycle often uncovers problems that can jeopardize the original project goals and deadlines.

The good news is that there are many options for testing and ensuring the performance and scalability of an application during the development phase. By focusing on performance early, many performance and scalability problems can be identified and eliminated before they ever make it into the final test phase (which is still very important).

In this chapter we will look at important performance-engineering methods that can be used during development. This includes dynamic architecture validation in the development environment, small-scale performance tests in continuous integration, and enforcement of development best practices to avoid common performance problems. We will also look at real-life examples showing how performance engineering in development and agile continuous integration have helped software companies maintain their agility while implementing new stories and improving overall quality.

In an upcoming section of this chapter, load testing—Essential and Not Difficult, we will discuss traditional performance testing and large-scale load testing, which remains important but can be streamlined using the techniques we discuss in the first part.

The performance methodologies we'll discuss are based on agile principles and are best integrated into the agile software process of continuous improvement. So we'll begin with a brief discussion of how this works and why it's a useful and important tool for software and performance engineers.