To estimate project duration we apply Celsius to Fahrenheit formula. $C$ is internal estimate and $F$ is what we tell PM: $C \times 9/5 + 32 = F$ days.

—DevOps Borat¹

In this chapter, we’ll examine the building blocks of DevOps. We’ll talk about the metrics, and you’ll learn that the cycle time is the most important metric for both development and operations. We’ll also discuss how to improve and accelerate software delivery. Let’s start with measurement and metrics.

**Measurement and Metrics**

A crucial aspect of software engineering is measuring what you are doing. Sooner or later, you’ll have to decide on which metrics you want to use during your software engineering process. You’ll have to consider which metric is meaningful enough to aid all participants, as well as the development and delivery processes.

Traditional projects emphasize measurement as an important tool for tracking progress, identifying the current status, and scheduling dates. Agile project settings try to find different approaches to create measurements, but often find themselves on dead-end streets when trying to bridge operations to development. Both traditional and Agile projects often emphasize the importance of measurement because you can only improve if you measure. Let’s take a brief look at how traditional projects understand measurement and metrics.

¹[http://twitter.com/devops_borat/status/111854984852811776](http://twitter.com/devops_borat/status/111854984852811776).
Traditional Use of Metrics

Classic metrics are often driven by numbers: they try to summarize and aggregate highly complex dependencies into single numbers. Counting leads to the illusion that we can understand something because we can quantify it. Have you encountered PowerPoint slides that provide one single number to illustrate the status of the project (e.g., cost effectiveness, capacity utilization, or meeting target scope, all in percentages)? Numbers suggest the illusion of control. However, numbers can be very misleading. Worse, counting all too often leads to perverse incentives. Even if managers are not assessing people based on process metrics, counting things affects behavior.

**Note** In classic projects, extensive measurement and metrics are often used to “manage by the numbers.”

Classic metrics, such as static code analysis or test coverage, may continuously draw the attention of the whole team without returning benefits in the same degree. It is simply too easy to fake metrics or to adapt the process to obey the metrics instead of improving the process itself (e.g., adding empty test bodies or commenting out broken tests to optimize the test coverage metric).

Conservative approaches that use “function points” to determine functionality can be misleading as well. You may measure the output of software development by using “functions points” to measure functionality, but you cannot derive the real value or any productivity from them.

Traditional metrics are often abused to compare teams or individuals. The best way to mess up the usefulness of any process metric is to use it to judge people. For example, if a manager uses the velocity or the number of defects to compare teams, the manager will have a serious problem.

**Note** Velocity is a metric that provides information about the “rate of progress” for the team. For example, velocity can be the number of user stories a team can perform in one interval, where it is important to include testing and shipping in the measure. Thus, a short form could be represented as “running tested features” (RTF).

Agile Approach to Metrics

Agile development methods require a disciplined approach to ensure that customer feedback, continuous testing, and iterative development actually lead to frequent deliveries of working, valuable software.

Software applications consist of functionality, and in many cases, new features will be created continuously. Only features that ship add value and form and improve upon a “solution.”

---

---

2 Different approaches for measuring capacity are discussed by John Allspaw in his book *The Art of Capacity Planning* (O’Reilly, 2008).