Reshaping the Iron Triangle: BETTER SOFTWARE PROCESSES FOR TOMORROW'S LIFE- SCIENCES TECHNOLOGY

According to some estimates, technology is advancing at a pace of 5 million new ideas/inventions per second. As new platforms and terms such as SaaS, Big Data, and cloud abound, there is exponentially increasing pressure for software producers and consumers to keep up with the latest use of software applications, mobile apps, or toolsets for mining social interaction. Staying technologically relevant in an industry that moves so fast is often a challenge, especially in the life-sciences industry, an industry that thrives on advancement but often struggles to keep pace with available and burgeoning technology.

Software is pervasive. It drives not only most of what we do in the life sciences but also much of what we do in our private lives. And although the latest apps or networked applications are all very exciting and cool, at the end of the day, it is all about software, and the process by which we develop, test, and deploy software is still the most important — and underappreciated — factor in successful on-time, under budget, and quality delivery.

Is process really that important?

In the past 50 years, software development methodology and toolsets have experienced a paramount shift. We've gone from mirroring construction efforts in highly structured physical environments — where designs are committed to before construction begins — to multiple, malleable methodologies that are better suited to handle evolving user requirements and provide the flexibility of iterative releases. Unfortunately, while other industries have embraced concepts such as rapid application development, the life-sciences industry has been sluggish in its adoption of faster-to-market processes for software development.

To decrease development time and costs while also increasing quality, it is imperative that organizations implement cutting-edge software development processes and toolsets.

Many companies that offer technology services face challenges on myriad levels —challenges that most often stem from a process standpoint. For example, juggling multiple, competing services efforts and needing to develop turnkey productized solutions require radically different management processes. In this environment, software development efforts tend to become tactical in their focus and reactionary in execution, rushing code changes into production where time is the primary motivator — and thereby placing quality at risk.

In project and software development management, there is a commonly known model called the Iron Triangle, which dictates that any effort is grounded by three opposing factors of cost, scope, and schedule — all of which drive the overall quality of the product. The concept behind the Iron Triangle model suggests that although a manager can increase attention to one factor, the other two will ultimately be detrimentally impacted. The problem lies in the fact that each one of these factors is important to different stakeholders within an organization or customer base —customers care about the scope, project managers care about the schedule, and financial managers care about the cost. The battle for top billing between these three factors is constant, but it must be recognized and dealt with, otherwise project failure is imminent. Identifying and implementing effective processes is the solution to managing the challenge of the Iron Triangle.

Enhanced processes allow an organization to scale back and grant less weight to the schedule and scope factors of the Iron Triangle, allowing more direct focus on controlling the cost of a software development effort (cost can be measured by a variety of factors including resource costs and budget). An organization's ability to do this is dictated by which software development process methodology it follows.

Contributed by

CHRIS MIDDLETON, *Vice President, Technology, MMG*

A software development process methodology is nothing more than a framework that dictates the process of constructing and delivering software. The process will typically include phases for design, construction, testing, and deployment. Many years ago, the most accepted waterfall methodologies dictated a finish-to-start order of events. For example, an organization would define software requirements to completion upfront, then construct the software, and finally deliver the software. Software development often takes a significant amount of time, and it was often only during the later phases of the waterfall methodology that stakeholders would realize that the laid out requirements would not work, were unneeded, or were implemented incorrectly — but it was too late to change them. In more recent years, organizations have developed and adopted newer development methodologies, whereby the focus is on faster construction and immediate stakeholder feedback and less on "big design up front" or BDUF.

Creating Agility

One of the more recent methodologies that many software organizations have embraced is Agile/Scrum — a process framework that is characterized by quicker delivery of smaller features. On a regular basis, an Agile/Scrum team determines what software will be built during a cycle of development known as a sprint, which lasts anywhere from one week to one month. During this timeboxed period, all planning development, testing, and deployment of the software takes place. At the end of the sprint, a review is held, and planning begins for the next period.

We have found great success in developing and executing a custom Agile software development process and coupling it with toolsets for continuous integration (CI). Combined, the two processes can drop the time to market for software deliverables considerably and provided a strong mechanism for managing stakeholder expectations. At the same time, increasing the level of code quality and procedural confidence.

The custom Agile/Scrum process differs from traditional Agile/Scrum models, which more commonly focus on a single product through its development lifetime, because of its applicability across mutliple product- and services-based engagements per month, all running concurrently. With a staff of fewer than 20 people dedicated to mobile apps, Web applications, Web services, and database services, the custom Scrum process supports roughly 300 work items, including new applications, features, or change requests, during a typical three-week sprint cycle. During this catch-all sprint period, a full release of an iOS mobile app may be tracked alongside Web-based API enhancements.

We shifted the Scrum product from a single focus to encompass every technology asset produced by the company, which is likely an offense to most Agile purists. However, the process has repeatedly proven its value by timeboxing a massive workload that is not only a mix of product and services, but also one that spans across industry segments. As a result, untenable tactical focus has been eliminated and the team has been elevated to more valuable strategic concerns such as product planning and innovation initiatives. In addition, through Agile/Scrum, the scope is better defined, monitored, and controlled through multiple, quick-to-market releases, which decreases the weight of the scope factor of the Iron Triangle.

How to Meet the Challenge of Delivering Validated Systems in an Agile Environment?

CFR 21 Part 11 validation is a very well intentioned assurance mechanism to ensure that healthcare software is created and delivered as intended. However, the validation development lifecycle was designed nearly 15 years ago, and adheres more to an older, less reactive processes than is advocated by Agile. Organizations have had modest success creating many of the artifacts necessary for a validation through automation, and when coordinated properly with an Agile approach, faster time to market can be achieved. Modularized systems, with proper planning and automation under CI can also aid in developing validated systems through multiple independent paths, reducing some of the lengthy phased finish-to-start process interpreted under the validated software development life cycle.

TO DECREASE DEVELOPMENT TIME AND COSTS WHILE INCREASING QUALITY, ORGANIZATIONS NEED TO IMPLEMENT CUTTING-EDGE SOFTWARE AND TOOLSETS.

A major time component to software development is fixing bugs and resolving conflicting code contributed by different developers. The CI process is a high-quality practice that reduces delivery time and increases software reliability. Through the use of CI, it is possible to minimize the time spent on tasks, such as build construction, testing, and deployment, and improve a technology organization's efficiency; this directly drives the ability to meet shorter timelines, letting an organization scale back the schedule component of the Iron Triangle.

What are the Real Benefits to CI?

CI enables the delivery of more reliable software, on a regular basis, in less time. A CI environment delivers considerable time savings, as once manual processes become automated. Automated testing reduces the number of software bugs and, downstream, lowers the amount of manual quality assurance testing that we need to apply. Also, by establishing automated deployment, software products are more quickly available for review or demonstration.

CI forces frequently integrating changes into an existing software base from multiple software developers and couples that with software build automation and automated testing. CI is currently a far cry from 15 years ago, when the ability for a team of developers to reliably share and update the same code base was sometimes a novel practice. Now, software can be edited by multiple people, changes can be integrated, tests against the changes can be automatically executed, and a build can be automatically deployed for review by a quality assurance team.

Unfortunately, many organizations inside and outside the life-sciences industry, fail to implement automated software testing or deployment automation and, as a result, are less likely to meet delivery timelines and run an increased risk of delivering broken software. When software is developed by a group, there is always a risk of not getting all of the individual contributions to work correctly and collaboratively as a whole. Traditionally, software contributions would be integrated (built) toward the end of an internal or external deployment release date — often resulting in a plethora of late nights and aggravation. CI forces code integration more often, spreading the aggravation of the build process across the entire development process, reducing overall risk of non-functioning integrated code.

According to ZDNet, global failures in information technology (of which software is a portion) is estimated at around $3 trillion in 2012 — out of an estimated $5.6 trillion total spent in 2011. Setting up a strong environment for controlling Agile development and CI takes considerable effort investment —something many organizations are not willing to undertake. However, when the benefits of reducing the scale behind the Iron Triangle's factors of schedule and scope, establishing an efficient process certainly becomes a worthwhile investment consideration.

Software development is a complicated practice. It requires extensive creativity but at the same time is grounded in hard math and engineering. An Agile development process, when mixed with CI, drives reduced software development costs and increased quality, while increasing strategic capacity — giving everyone more time to keep pace with the new technologies and their real impact on software in life sciences. PV

***MMG** is a full-service global health communications group that specializes in patient recruitment and retention.*

For more information, visit mmgct.com.