



SCR

TO TACKLE SOFTWARE DEVELOPMENT USING HIGH-IMPACT SOFTWARE DEVELOPMENT



UM
CT TEAMWORK
EVELOPMENT PROJECTS

by Laszlo Szalvay

Today, companies find themselves in the midst of a business world that is continually shaped and reshaped by emerging technology. From Web 2.0 to cloud computing, these new technologies are leading companies to completely reevaluate how they operate and interact with customers. And no one better understands the opportunities and risks associated with a rapidly evolving technology landscape than companies from the tech sector—particularly the software development industry. In short, the ability to respond quickly and flexibly to new business conditions is integral to survival. With that in mind, software developers require a new approach to project management that is nimble enough to keep pace in such a chaotic business climate, but determining which management framework will yield the best results isn't always easy.

One of the new methods currently being employed with success is Scrum, an agile method of project management that uses iterative, incremental work cycles known as sprints to provide frequent opportunities to assess and revise direction. While Scrum's iterative nature ensures that teams are always mindful of the big picture, the framework also facilitates ongoing communication and collaboration to yield high-impact teamwork. When people are connected through frequent check-ins, it enables teams to improve their processes in the long term by becoming the sort of groups that can achieve the near impossible. When an organization develops several of these hyper-performing teams, it creates a path toward sustainably lean operations. In other words, teams like this can do more with less.

An Overview of the Scrum Framework

Before explaining how the Scrum framework achieves all this, it's helpful to begin with what Scrum is and how it differs from other approaches to software development. To begin, Scrum can be defined as a simple management framework for incremental product development. Development is performed by one or more cross-functional, self-organizing teams of about seven people each. These teams use fixed-length iterations called sprints, which are typically fourteen to thirty days in length. Unlike waterfall, the historically dominant method of software development, Scrum

does not assume that all of a project's requirements can be known at the outset or that development can occur in a linear, "single-pass" fashion. Instead, Scrum assumes that more information about the desired product will be collected during development and, to accommodate that, proceeds in sprints, allowing the team to revisit areas of development throughout the lifecycle.

The Scrum method is deliberately designed as a framework—i.e., a lightweight management wrapper that can be applied to existing processes. However, every part of Scrum's minimal framework is essential for realizing its core tenets of facilitating productivity through communication, collaboration, and self-organization. Given its spare structure, it's critical that all of Scrum's roles and processes are observed. Here's a quick overview of Scrum's primary roles and meetings.

SCRUM ROLES

- The *product owner* constantly re-prioritizes the product backlog to reflect those items with the highest business value. This individual is responsible for communicating product vision to team members and negotiating sprint goals with them each sprint. He also is responsible for yielding a return on investment and therefore possesses the authority to accept or reject each product increment.
- The *Scrum team* is a cross-functional and self-organizing team

of about seven members that is responsible for delivering a functional product increment each sprint.

- The *ScrumMaster* facilitates team productivity and self-organization by removing impediments that obstruct progress, enforcing Scrum's rules, and ensuring that all Scrum artifacts are highly visible.

SCRUM MEETINGS

- During the *sprint planning meeting*, the product owner and the team negotiate the work that team members will attempt to complete in the next sprint. The product owner is responsible for identifying which work is of the highest priority. Likewise, the team is responsible for committing to the amount of work it can accomplish.
- The *daily scrum meeting* allows team members to deliver updates to one another on a daily basis. Every day, at the same time and place, team members spend fifteen minutes reporting to one another. Each team member reports to the rest of the team what he did the previous day, what he will do today, and what impediments block his progress.
- The *sprint review meeting* occurs at the end of the sprint. At this meeting, the team demonstrates the functional product increment it has developed and the product

owner either accepts or rejects the work, based on the previously negotiated agreement. This is an opportunity to “inspect and adapt”—that is, to examine the product’s progress and revise direction, if necessary, for future sprints.

- At the *sprint retrospective*, the team inspects and adapts its own processes. During this meeting, the team reflects upon its performance in the past sprint and brainstorms ways to improve going forward.

A Mix of Structure and Flexibility Empowers Teams

Scrum’s iterative and incremental approach to development benefits teams. During each sprint, a team commits to a defined amount of work, which it must complete over the course of the sprint. These action items—called *product backlog items* in Scrum—are negotiated between the team and its product owner. However, once work is assigned for the sprint, the product owner cannot give the team additional work until the sprint concludes and planning takes place for the next one. The team is empowered with the autonomy to determine how it completes the work and in what priority it will tackle it. In Scrum, this is called *self-organization*.

For traditional project managers who try Scrum, resisting the temptation to micromanage can be a considerable challenge. Learning to trust that a team will self-organize effectively to complete its sprint goals may be even harder. But this aspect of the Scrum framework carves out a stable, uninterrupted stretch of time for teams to complete work. Given the inherent volatility of writing software, Scrum endeavors to minimize the thrash of the variables it can control—in this case, the disruption of an overly demanding product owner. Moreover, the iterative nature of sprints allows teams to establish a rhythm, creating a kind of routine velocity for each sprint.

As team members become increasingly comfortable with one another over time, they learn how to work together more efficiently. Such hyper-productive teams are the key to organizations’ maximizing their potential.

In another departure from the waterfall method of project management, Scrum possesses the unique characteristic of allowing teams to begin work on a product without knowing all of its requirements. Where waterfall historically asserted that all requirements could be known absolutely at the beginning of a project, Scrum believes that, just like work, requirements can be added incrementally as development progresses. With waterfall, teams waste time and energy attempting to identify every feature of the product before building begins. Not surprisingly, they typically fall into a state of “analysis paralysis,” in

“Given the inherent volatility of writing software, Scrum endeavors to minimize the thrash of the variables it can control...”

which the requirements-gathering phase becomes so daunting that the team simply does nothing at all. Scrum, on the other hand, knows that defining certain features may only be possible once other, more basic, functionality has been built.

Of course, Scrum still asks teams to produce a functional product increment each sprint, which means that, even during that initial sprint, the team must settle on some foundational functionality to build immediately. That first sprint forces teams to make hard decisions and begin work, thereby saving the time and money that would have been wasted futilely trying to wrap their arms around as-yet-unknown requirements. In software development, this is a radical step toward responsibly conserving time and money. By acknowledging the limitations of what it can know at the outset, the development team is able to focus on those requirements it is certain the client expects. Thus, the team immediately begins building essential functionality, adding other features to

the product scope through each meeting with the customer.

How Scrum Generates Business Value

One of the greatest advantages of Scrum over traditional project management models is that Scrum projects are developed iteratively and incrementally. Because teams work in repeatable, timeboxed work cycles, they add functionality to products with each successive sprint. To ensure that the team progresses toward completion, Scrum mandates that teams produce a “potentially shippable increment” every sprint. This means teams always have a demonstrable product to show a client. It might contain only barebones functionality at first, but it forces teams to make tough decisions and hit the ground running. Equally important, it produces a work in progress to share with the customer, who can provide direct feedback. In fact, there’s a strong business case for building frequent customer interaction into the Scrum process. Customers typically

need to interact with a piece of software to better articulate their vision for it. This phenomenon is called “I’ll know it when I see it,” or “IKIWISI” for short.

Of course, the true test of any project management strategy is its ability to generate business value. Scrum accomplishes this in a number of ways, all of which contribute to the end goal of creating a product that meets the customer’s expectations. As stated above, Scrum brings the customer and the development team together as often as every sprint. Because the customer is involved throughout the development process and is consulted once per iteration, the team can never deviate very far from the customer’s vision. Even if, for example, a development team takes the product in the opposite direction, it can’t get too far off track because the sprint is timeboxed. Compared to waterfall development, in which development proceeds in a linear, unchecked fashion, Scrum’s abbreviated feedback loop guarantees that the team always circles back to the customer to



...organizations must leverage Scrum's emphasis on

transparency and communication

to manage employee anxiety about the transformation.



check the team's progress against customer expectations during the sprint review.

Because requirements are added incrementally in Scrum, it is possible to delay certain decisions until the last responsible moment. This provides customers with the flexibility to adapt—even to late-breaking circumstances—without breaking the bank. Imagine a competitor rushes to market a rival product that includes new functionality absent in your product. Even though your product is nearly complete, its scope can quickly be revised to incorporate this feature. With the product's base functionality already in place, another feature simply represents another increment of work. By contrast, waterfall's exhaustive, up-front requirements would have locked the team into developing a product that could not compete in the marketplace. After all, this is why Scrum believes requirements should be gathered throughout the development process. This ability to make last-minute revisions to scope at a low cost is what makes Scrum so "agile." Even when it appears that a competitor's product has already won, Scrum allows developers to respond quickly, nimbly, and inexpensively by extending the product lifecycle by a few iterations. Instead of a product that is rendered irrelevant by the market, the team simply has an underdeveloped product, with the additional functionality, which can still go to market and compete.

Challenges

While Scrum addresses many of the shortcomings present in other development methods, it does present a few challenges of its own. However, the majority of these impediments are limited to

the transformation process, in which an organization first transitions to Scrum. Because Scrum asks teams to adopt drastically different working habits, it is often met with a significant culture clash. It is not uncommon for individuals to fear change, especially in the workplace, and the effects of change during a Scrum transformation can range from mild concern about learning to work differently to an outright refusal to comply. Therefore, organizations must leverage Scrum's emphasis on transparency and communication to manage employee anxiety about the transformation.

In addition to some cultural resistance, organizations commonly observe some administrative roadblocks, as well. Given Scrum's emphasis on teamwork and the relatively small number of roles it contains, human resources departments must revise existing career paths and incentive programs to ensure that the organization's values align with Scrum's. The good news for organizations facing the challenge is that Scrum's incremental approach to development can be applied to situations like these, as well. Often, a "transformation team" is utilized to make sure that Scrum's principles and processes are implemented at both the team and organization levels.

Conclusion

In the end, Scrum succeeds because its emphasis on communication brings people together to share problems, brainstorm solutions, and realize a vision. Fueled by this kind of engaged collaboration, teams form bonds through shared commitment and past successes, which, in turn, push teams to levels of productivity they never dreamed they could achieve. To facilitate this communica-

tion, Scrum believes that the customer, who understands what the product should become, and the development team, which performs the work, should meet frequently to make sure that they build the *right* product. Because a product's lifecycle is divided into increments, an opportunity to pause and assess the product's direction against real-world conditions is never more than a single sprint away. And though traditional project managers may be frightened by Scrum's belief that development can begin before all requirements are known, they shouldn't be. In fact, this approach to development is actually rooted in reality, taking inventory of empirical data to help the team make informed decisions as information becomes available. Waterfall effectively ignores the fact that, by the time a product is ready to ship, the thrash of the real world has rendered a team's work irrelevant. Scrum actively guards against this wasted effort by encouraging teams to dive in and build requirements as they become known.

When a team's activities are determined by customer feedback and empirical data, there's a greater likelihood that the team will be able to proceed with confidence and clarity. And when teams can progress with such measured intentions, they often outperform their own expectations. Especially for tech companies, who must remain attuned to how new technology impacts their business, Scrum's combination of frequent communication and attention to real-world factors enables teams to build products that customers actually want, while operating in a sustainably lean fashion. And that translates to success—no matter what business you're in. **{end}**