WHITE PAPER

The Five Most Common Gaps in Scrum Adoptions

Jenny Stuart, VP Consulting, Construx Software
Erik Simmons, Senior Fellow, Construx Software
Melvin Perez, Senior Fellow, Construx Software

Version 1.0 2, October 2018

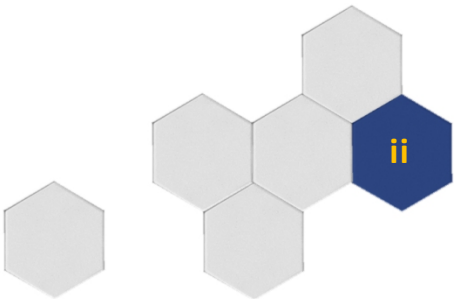
Scrum is the most widely used Agile framework in the software development industry today. One of Scrum's advantages is that it provides a significant amount of structure. The framework includes specific roles, events, and artifacts that work together. It aspires for transparency, self-empowerment, and empirical process control.

Construx works with many organizations across a diverse set of industries. We find that many teams encounter a consistent set of challenges with Scrum due to similar gaps in their Scrum adoption. This paper describes those gaps and shares resources that will help you address them.

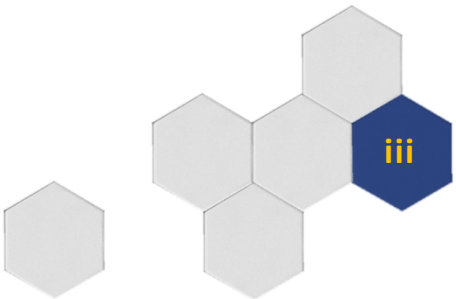
Construx

Copyright

Construx Software Builders, Inc.
10900 NE 8th Street, Suite 1350
Bellevue, WA 98004
U.S.A.



Contents



#1 Missing or Insufficient Focus on Quality

The Scrum Guide states that Scrum Team members must have a shared Definition of Done (DoD). However, it provides no specific and concrete guidance on what should be in your DoD. The Scrum Guide is intentionally vague because the details do, and should, vary widely from team to team.

Many of the Scrum teams that Construx works with do not have a shared Definition of Done that everyone understands, agrees to, and uses. Some common anti-patterns Construx sees are:

- Missing Definition of Done. Many teams Construx encounters have never established a Definition of Done.
- Ignored Definition of Done. The team has a definition, but it is not used or referred to during the Sprint to determine whether an item is ready for the Sprint Review.
- Weak Definition of Done. A Definition of Done that omits quality or addresses it only superficially. For example, development considered complete without unit testing, functional testing, any documentation, and other elements necessary for high-quality work.
- The Mandate from On High. Someone somewhere in the organization provides a Definition of Done without consulting with the teams. The imposed definition often contains inappropriate items and items that individual teams are incapable of completing within the Sprint boundary.

A Definition of Done should state the teams' expectations for the following:

- Testing (unit, integration, and system)
- Reviews (requirements, design, and code)
- Automation
- Tool use (static code analysis, security checks, and code style checks)
- Documentation[1] (as-built architecture specifications, design documentation, user documentation, release notes)
- Deployment infrastructure readiness (release notes, deployment scripts)

[1] Additional information on incremental documentation creation can be found in Construx's ***Agile Documentation Practices*** white paper.

Each team should establish definitions that are unambiguous and measurable—such as "*At least 70% unit test coverage for new code*"—and put in place mechanisms to monitor them.

The Definition of Done should bring the software produced within a Sprint to a potentially releasable state or as close as the team can possibly get to that state. Any items preventing deliverables from being potentially releasable should become process improvement items for the team and organization.

Figure 1 shares a resource that discusses the Definition of Done and provides an example.

Figure 1 *Resources that Support Improving Quality*

Definition of Done Practice Paper

Please contact Construx for this resource.

#2 Insufficient Staffing of the Scrum Roles

Scrum defines three roles: Product Owner, Scrum Master, and Team Member. Each of these roles has a unique set of defined responsibilities that complement the other roles' responsibilities. In combination, these three roles provide checks and balances. These checks and balances should drive evolutionary improvement by exposing and resolving issues that impede team efficiency and effectiveness. The common anti-patterns Construx sees are:

- Missing Product Owner. The Scrum Team attempts to work without anyone in the Product Owner (PO) role. As the Scrum Guide states, "The Product Owner is responsible for maximizing the value of the product." A team without a PO lacks sufficient guidance on what the business and stakeholders need. Because the PO guides the work of a team of up to nine people, it is a vital role for organizations to staff.
- Inactive Product Owner. Someone is named as the PO but does not participate with the team sufficiently, if at all. This is the same general pattern as the missing Product Owner, but in this case, the team can identify a specific person as PO.
- Proxy Product Owner. The right person doesn't have the time to be, or interest in being, a PO, so someone else is designated a Proxy PO. This can work, but it is a rare exception when it works well. In general, the Proxy PO cannot provide the necessary detail and day-to-day information necessary for Scrum to be successful.
- The "Oh By the Way" Scrum Master. Many teams have a Scrum Master who was the person most willing (or even least unwilling) to take on the role. Sometimes this is better than having no Scrum Master, but usually these people aren't interested in becoming a great Scrum Master. Rather, they are interested in being

great developers, testers, or technical writers. Also, they are often given the guidance that a Scrum Master's only role is to run the meetings. A great Scrum Master who understands the role brings much more than that to the team, the PO, and the organization.

- Command and Control Scrum Master. A command and control Scrum Master will tell the team what to do rather than enabling them to become a self-empowered team, thereby greatly impeding learning.

For Scrum to be successful, it is crucial to staff the Product Owner and Scrum Master roles appropriately. Organizations are rarely in a position to hire new staff when they transition to Scrum. Instead, it is a matter of identifying how the organization can best fill each role by using its existing people.

Figure 2 outlines a set of resources that can help you fully staff the Scrum roles and provide the professional development for those individuals to be successful.

Figure 2 *Resources for Staffing Scrum Roles and Developing the Staff*

Staffing Scrum Roles White Paper

Product Owner Professional Development Path

Scrum Master Professional Development Path

These resources are publicly available.

#3 Insufficient Backlog Refinement

The Scrum Guide states that "Product Backlog refinement is the act of adding detail, estimates, and order to items in the Product Backlog," and it says that this process should consume no more than 10% of the team's time on an ongoing basis. Although backlog refinement is not a defined event in Scrum because there is no set time in the Sprint at which it must occur, it is crucial for successful Scrum work.

Insufficient backlog refining makes everything in Scrum harder. The Product Backlog is the fuel for the Scrum engine. Without good fuel, teams have a hard time working effectively and efficiently. Common symptoms include:

- Sprint Planning takes longer than necessary. One team described Sprint Planning as taking two days for a two-week Sprint, which is far too long. Sprint Planning should focus on the *how* we will build it because the *what* we're building has been sufficiently understood in refining. Good refining ensures Sprint Planning is a focused, efficient, and effective event.
- There are excessive questions about the details of the work during the Sprint. Some teams have described being unable to complete work in a Sprint because they did not know or understand important details of what they were building. Scrum relies on just-in-time conversations rather than detailed, speculative

specifications, but when these conversations become excessive, the team cannot make rapid forward progress.

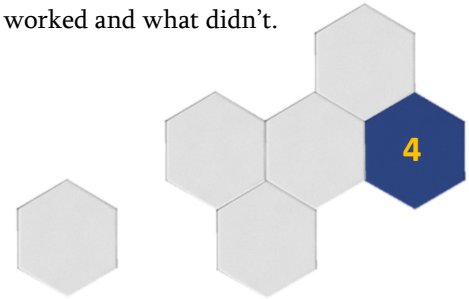
- Work consistently slips into the next Sprint. Many teams bring in stories that are too large to be brought to their Definition of Done within a Sprint. Instead, sufficient refining must be done for large stories to be decomposed into items that are small enough to fit in a Sprint.
- The team is unable to do any parallel work. Scrum is a "team sport," and a good Scrum team attacks work in parallel. If, for example, the testers cannot build test cases in parallel with developers writing the code, this is often because the team lacks a sufficiently shared understanding of what they need to build. Refinement must yield a *clear*, *common*, and *coherent* understanding of the work to be done in a Sprint.
- The team experiences excessive downstream rework. When lots of required changes are identified in the Sprint Review or after the Sprint is complete, it is usually a sign that the work was not well understood by the entire Scrum Team. Change should occur before items enter the Sprint, during refinement. It is expensive to rework an item, sometimes multiple times, before getting it right. As mentioned above, Scrum is designed for conversations and flexibility—Construx encounters many teams that could vastly improve their performance by reducing easily preventable rework through better conversations and backlog refinement.

Construx recommends that teams spend one hour twice a week refining the Product Backlog. Two one-hour sessions provides the best balance of flexibility and time to focus on the task.

#4 Missing or Ineffective Retrospectives

The Sprint Retrospective is a process and teamwork inspection event—a powerful tool that can make Scrum teams great. But many teams have retrospectives that do not generate significant improvement, and some teams stop holding retrospectives when they discover, correctly, that poor retrospectives are a waste of time. Common issues that reduce or eliminate the benefits of retrospectives include:

- Failing to address core issues. The important issues are never exposed, discussed, or addressed because of cultural barriers, interpersonal issues, organizational politics, line management reporting conflicts, or other reasons. The changes identified might address some symptoms of the underlying problem, but that problem is never solved.
- Failing to identify a change the team will make in the next Sprint. Holding a retrospective without identifying an improvement misses the entire purpose of the retrospective. It is not sufficient to talk about what worked and what didn't.



Starting with the 2017 revision, the Scrum Guide *mandates* that teams identify at least one high-priority improvement during a retrospective for inclusion in the next Sprint Backlog. This change points to the frequency of teams failing to hold retrospectives that result in immediate improvement actions.

- Identifying vague improvements. Actions such as "We should do better refining" or "We should meet our Sprint commitments more often" are hard to implement because they are so broad and unmeasurable.

  Teams should take these vague improvement ideas and brainstorm specific, measurable changes that they can try so as to improve. For example, a team might state, "Let's update our Definition of Ready to include 'No stories larger than an eight can be brought into a Sprint,'" when it believes that the size of stories is causing them to spill over from Sprint to Sprint.

- Holding the same retrospective over and over and over. The most common retrospective technique is asking people what worked and what didn't work, often using the "go around the table" approach. Any technique used repeatedly quickly becomes boring and is unlikely to generate unique, valuable insights. Scrum Masters should utilize a number of different retrospective techniques to keep team members engaged and to support an open, safe environment.

- Not looking at past performance. It is useful to mine the team's historical data and share it in retrospectives to help the team see patterns. Data like the three-Sprint moving average velocity, initial total hour estimates vs. actual, % of Product Backlog Items committed vs. delivered, and so on can be illuminating.

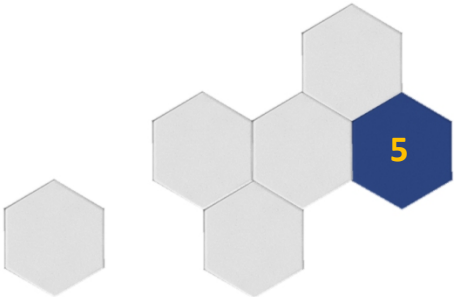
- Never or rarely making the changes identified in the retrospective. An identified change can be impactful only if the change is actually made. Teams can add action items related to the change to the Sprint Backlog so that they are visible and so that any work necessary to implement the change can be estimated and resourced.

- Agreeing to a laundry list of issues. Teams should identify one valuable change, perhaps two, that they will make in the next Sprint. Agreeing to a long list of changes generally means that no change will be made.

Retrospectives should be fun and energizing, and they should result in continuous improvement via Scrum's *inspect and adapt* mantra.

#5 Lack of Focus on Incremental Value Delivery

Many teams throughout the industry use Scrum, but few of them create potentially shippable software at the end of each Sprint. Teams that do create shippable software for each Sprint often have disconnects with their stakeholders that have prevented the team from delivering what is most valuable first.



Scrum places a lot of responsibility on the Product Owner to understand the team's stakeholders, discover their needs, properly order the Product Backlog to bring the highest return on investment to the business, and convey the details to the team so that they can quickly and continuously deliver stakeholder value.

Some common problems that Construx sees often are:

- Missing key stakeholders.
- Failing to deeply understand what the stakeholders value.
- Not staying current with changing stakeholder values.
- Not exposing key assumptions in decision making and prioritization.
- Breaking large pieces of work down in ways that don't deliver value incrementally. For example, major epics given to component teams that don't deliver customer value until each epic is completed by every single team.

Scrum is an excellent delivery engine. The Product Owner and Product Backlog are the steering mechanism for that engine. A significant amount of work is necessary to ensure that the right things are placed at the top of the backlog.

Figure 3 outlines a set of resources that can help you understand, capture, and communicate value to and within the Scrum team.

Figure 3 *Resources for Understanding Value and Sharing it with the Team*

Value-Driven Delivery White Paper

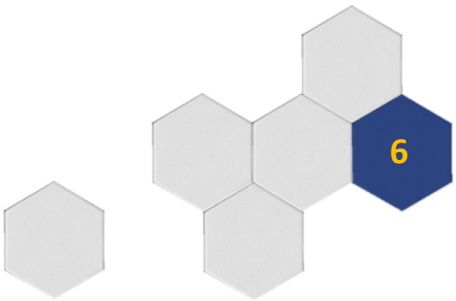
Landing Zone Practice Paper

Product Owner Career Path

Requirements Decomposition Practice Paper

Refining Practice Paper

The white paper and "Product Owner Career Path" are publicly available. Please contract Construx for the practice papers.



Contributors

Jenny Stuart, VP Consulting

jenny.stuart@construx.com
+1(425) 636-0108

Erik Simmons, Senior Fellow

erik.simmons@construx.com
+1(425) 636-0100

Melvin Perez, Senior Fellow

melvin.perez@construx.com
+1(425) 636-0120

Construx

Construx Software is the market leader in software development best practices training and consulting. Construx was founded in 1996 by Steve McConnell, respected author and thought leader on software development best practices. Steve's books *Code Complete*, *Rapid Development*, and other titles are some of the most accessible books on software development with more than a million copies in print in 20 languages.

Steve's passion for advancing the art and science of software engineering is shared by Construx's team of seasoned consultants. Their depth of knowledge and expertise has helped hundreds of companies solve their software challenges by identifying and adopting practices that have been proven to produce high quality software—faster, and with greater predictability. For more information about Construx's support for software development best practices, contact us at consulting@construx.com or call us at +1 (866) 296-6300.