BEST PRACTICES WHITE PAPER

10 Keys to Successful Scrum Adoption

Jenny Stuart, Vice President of Consulting, Construx Software

Version 2, November 2011

Contributors

Earl Beede, Senior Fellow

Jerry Deville, Senior Fellow

Eric Rimbey, Senior Fellow

Melvin Perez, Senior Fellow

Scrum is an approach for Agile software development and is the most commonly adopted Agile approach in the industry today. While Scrum has core principles that must be implemented to effectively adopt the approach there are a number of ways that it can and should be modified and expanded to support the unique needs of individual software development organizations. Construx has worked with hundreds of organizations to implement Agile approaches including Scrum. From this work we have identified ten keys that help an organization to identify the changes necessary to ensure that Scrum meets its unique needs, goals, and challenges.

Contents

1. Evaluate Scrum's Suitability

Scrum is an iterative, incremental method for managing the top-level workflow at the team level. It is one of the Agile software development approaches which also include Extreme Programming (XP), Feature Driven Development (FDD), and Crystal.

At its core, Scrum uses fixed-length iterations (Sprints), has time-boxed daily Scrum meetings, maintains work items in a Product Backlog, and completes work small batches via a Sprint Backlog. Scrum includes continuous improvement feedback loops for both the product and the process itself. It has three predefined roles: Scrum Master, Product Owner, and the Development Team.

Construx has worked with numerous organizations in a wide variety of industries to implement Scrum on specific projects or throughout the entire organization. In our experience, Scrum is often, but not always, a good fit for an organizational or project software development approach. Software projects vary quite a bit in the stability of the requirements and user needs, the number of different geographies involved in the development effort, and the level of access to end customers. Because of these and other variations, it is important that the life cycle or methodology fits the specific needs of the project.

Therefore the first key to a successful Scrum adoption is to determine whether or not the methodology is appropriate for a project or for most of the projects in the organization. Aspects to consider include the following:

- **The accessibility of the customers or a representative of the customers.** The incremental delivery of working functionality that is accepted by the end customers or their proxy is fundamental to Scrum. It is important for successful adoption of Scrum that the Development Team has access to the voice of the customer throughout the entire project.
- **The level of technical talent in the organization.** Scrum works best with experienced, disciplined developers who can guide the technical evolution of the system. Skilled developers can effectively balance the need to produce working software in the near term with long-term architectural implications. They can provide the guidance necessary to ensure the system can be extended over time with additional functionality and achieve the desired level of performance and scalability. Without the necessary baseline of technical expertise, there is a significant risk that the system that is developed will be brittle, difficult to extend, and unable to achieve the desired level of performance.

- **The stability of the requirements for the system.** When the requirements for a system are well understood and stable, it is certainly possible to use Scrum, other incremental approaches may be as or more suitable. With stable requirements, an approach such as Staged-Delivery can sometimes be a more effective approach. Scrum is a better fit for projects that require more exploration of the users' needs. Its incremental commitment approach and iterative delivery are highly effective in ensuring that users' needs are communicated to the team and that there are numerous checkpoints exist to confirm the right functionality is being delivered. With well-understood and stable requirements, Scrum can add overhead that is not needed for a successful outcome.
- **The amount of geographic and time zone distribution.** Scrum was designed with a small, co-located, cross-functional team in mind. While it is possible to adopt parts of Scrum or extend Scrum to support geographic and time zone distribution, using Scrum is significantly more difficult to use with a widely distributed team.
- **The supporting infrastructure.** Iterative development requires that supporting infrastructure is in place or can be built during the project. Support infrastructure includes the ability to perform daily or continuous builds, perform automated validation of the builds, conduct automated unit testing and regression testing, and promote and release builds.

Keeping these considerations in mind will help determine if Scrum is the right fit for an individual project or for use throughout the organization. In cases when a full Scrum adoption is not a good fit, it is generally still possible to adopt specific Agile practices and techniques on the project that will allow it to deliver functionality in a more incremental and iterative approach.

2. Commit to Core Principles

It is critical that the core principles of the Scrum methodology are understood and then adopted by teams. Many of the project failures that Construx has reviewed occurred because of the lack of focus on the core principles. The outcome of these projects has ranged from outright project cancellation to systems that needed to be built again from the ground up because the underlying infrastructure could not support the long-term needs of the project to projects that were delivered late, over budget, and had numerous features that did not meet the users' needs.

As a team or organization adopts Scrum, it is important that it focus on implementing the following core principles:

- **Steady Flow, Time Boxing.** Scrum has short, fixed length Sprints. Sprints typically last for two, three, or four weeks. The use of these fixed-duration Sprints supports ongoing progress visibility and enables the team to calibrate its estimation approach and estimates. The short Sprints ensure that the team and project stakeholders pri-

oritize the functionality and break the system down into small, value-added elements.

- **Maintain Quality.** The Development Team delivers working functionality that is in a "potentially releasable state" at the end of each Sprint. Each Sprint needs to deliver something that is useful to the customer. During initial Sprints, the deliverable can be the creation of a thin vertical slice of working functionality that demonstrates a very simple capability of the system. In later Sprints, the deliverables are the incremental delivery of features from the Product Backlog.

  Bringing software to a near releasable state commonly means that the Sprint includes the detailed requirements work, design work, construction, unit testing, and system testing of the functionality allocated to that Sprint. One common mistake is to postpone system testing of the functionality to the next Sprint. While full performance, load, scalability, and other aspects of system testing can be completed in later Sprints, it is critical that the system be tested from end to end within the Sprint so that the functionality can be validated.

  The final definition of what it means to bring the system to a near releasable state should be clearly established by each Scrum team. Establishing a shared understanding of what it means to be done with each Sprint and with each feature within the Sprint is a fundamental necessity to successful Scrum adoption.

- **Focus on Value**. The working functionality that is delivered at the end of each Sprint reflects the current priorities of the customers. While the team is responsible for committing to the feature set it can complete in the duration of the Sprint, the end customers—through communications with the Product Owner—are responsible for determining the specific features they perceive as providing them with the most value. The use of short iterations means that the aspects of the desired functionality that have the highest value are delivered early in the project. Benefits of short iterations include opportunities to provide early releases of functionality to customers and the ability to modify the requirements throughout the project if the users identify higher value functionality.

- **Empowered Teams.** Scrum results in a self-organizing, self-empowered team committed to achieving the goals of the project. A team that collaborates openly and freely, and that includes all team members in the effort, is critical to Scrum. The approach makes greater use of verbal communication than written communication, which makes having a cohesive team with good rapport critical to success. One major difference from teams in more traditional project life cycles is that all members of the team communicate openly and frequently about items such as roadblocks, status, and upcoming tasks. One sign of a solid working team is that the daily Sprint meetings do not resemble traditional project status meetings; rather, they feature communication between all team members. This communication takes the form of peer-to-peer discussions rather than discussions between individuals and a project manager.

- **Continuous Improvement.** Short Sprints provide ongoing opportunities for the team and organization to learn from their experiences and make changes that increase the likelihood of success. Typically issues with the implementation of Scrum on a project or Scrum's interaction with other organizational processes occur during the first Sprints. During the first two or three Sprints it is critical that the team continue Sprint retrospectives or their equivalent to identify opportunities to improve the approach on the next Sprint. These retrospectives enable process tailoring to ensure that Scrum meets the unique needs of the project and organization. Retrospectives after each Sprint provide ongoing opportunities to streamline the approach and improve the project.

There are numerous ways to tailor Scrum; however, this does not change the importance of adopting the fundamentals. For a successful adoption of Scrum on a project or within an organization, the fundamentals of the approach must be understood and implemented by the team and its stakeholders.

3. Refine the Scrum Adoption

While by-the-book Scrum can be a good fit for some project teams, more often adopting organizations need to tailor the methodology to meet the needs of their culture, to fit within larger organizational constraints, and to meet the needs of specific projects.

In Construx's experience it is important to begin with a by-the-book Scrum adoption to ensure the fundamentals are understood and the approach is adopted. From there while it is important not to violate any of the core principles of the methodology, refining Scrum is quite appropriate in a number of areas.

Some of the common ways that Construx sees organizations or teams tailor Scrum include the following:

- Establish a Sprint length that works for each project. Teams often decide to shorten the duration of Sprints from one month to three weeks, or even two weeks. Construx has also worked with teams that have varied the duration of Sprints during the project. This decision is typically made after a Sprint retrospective, and the duration is changed for the next Sprint. It is important that this change does not occur within a Sprint, as that can reduce visibility into project progress and obscure product quality or process adoption issues. It is also important that the Sprint length is not constantly changing as the team's velocity will change significantly when the Sprint length changes.
- Incorporating technical practices from XP, FDD, Crystal, or more traditional development approaches to support effective product development for the project or within an organization. Scrum is a management framework for software development; it does not specify the details of the software development practices that are used. As Scrum is adopted, it is important to determine the specific design, con-

struction, and testing practices that are necessary to deliver software given the unique needs, constraints, and goals of the adopting organization.

- Assigning multiple individuals to the role of Product Owner. This change to Scrum is acceptable when a single individual cannot be available for the duration of the project or when the project has different individuals for strategic product guidance and for the tactical decision making necessary to achieve the long-term product objectives. When considering this modification it is critical to ensure it is being done because it is right for the project or product. It should not be done because the organization or team wishes to avoid hard decisions about the impact of the new roles on the current organizational roles of product manager, program manager, and project manager.
- Establishing project and/or organizational-wide "Definition of Done." One common way that Agile teams ensure they have a focus on quality is to have a clear Definition of Done and have all teams formalize their definition. Organizations often create a framework that includes the specific areas that should be considered by each team or create a minimal Definition of Done that teams can extend as needed.
- Adding quality assurance, reporting, or other processes necessary to address external constraints such as regulatory compliance. Changes related to this category include incrementally building required documents throughout the Sprints, adding personnel on the project, or building infrastructure to comply with organization processes.
- Establishing mechanisms for tracking Sprint and overall project/release progress visibility. Many organizations implement Sprint Burndown Charts, Release Burnup Charts, and collect velocity information to collect and display this information.

Beyond expanding upon the Scrum methodology, organizations occasionally need to map Scrum to larger organizational processes such as a phase-gate or other project governance processes. See Construx's White Paper *Introducing Agility into a Phase-Gate Process* for more information.

4. Transform Roles

Scrum introduces the three major roles of Scrum Master, Product Owner, and the Development Team, which are outlined in detail in Table 1. As part of adopting Scrum, it is important to understand the implications of implementing these new roles. While these roles are in some ways similar to traditional software development, there are subtle but important changes to the roles. One example is that is that while a project manager may fill the role of scrum master, some of the traditional activities such as the assignment of tasks are not part of the role of a Scrum Master.

Table 1 *Scrum Roles*

Role	
Scrum Master	The Scrum Master supports the Scrum process and facilitates team activities. Scrum Masters ensure the process is being followed, enable cooperation between team members, help the team reach decisions, surface dependencies, remove barriers, and guard the team from outside interference. The Scrum Master, as the gateway to the technical department, provides visibility into the team's progress by using artifacts such as a Sprint Burndown Chart and a Release Burnup Chart, and providing status to the technical stakeholders. Keys to success in this role include using facilitation and coaching rather than traditional command-and-control project management practices. The Scrum Master needs to create a shared sense of purpose and an environment that encourages the whole team to connect and to contribute to the project. Without a strong sense of team commitment, the team members will act as independent contributors rather than coming together into a fully functional Agile team.
Product Owner	The Product Owner defines the features of the product, orders/organizes the features according to the business value, works with the Development Team to define the feature set for each Sprint, removes barriers, and accepts or rejects the outcome of the Sprint. The Product Owner, as the gateway to the business side, works with the business to create user stories, defines acceptance criteria, and clarifies stories by conducting discussions with the Development Team and bringing in Subject Matter Experts as necessary. Keys to success in this role include the ability understand and communicate business priorities and requirements to the team with minimal delays. The Product Owner needs to be a steward for the business community. Once the Sprint has been committed to, the Product Owner is not permitted to change the content of the Sprint unless there is a rare scenario that justifies stopping the Sprint and restarting with a Sprint planning session.

Role	
Development Team	The Development Team is responsible for determining the set of work that can be completed in a Sprint by creating Sprint Backlog items from the prioritized Product Backlog. The entire team is responsible for delivering a potentially releasable increment of work that confirms to an agreed upon quality bar. At the end of each Sprint, the Development Team demonstrates to the Product Owner that it has successfully completed the work. For example, they have completed stories that fulfill the story acceptance criteria and meet the agreed upon "Definition of Done". Keys to success include having a small (5-9 person team) and ensuring that the team works together to meet the Sprint goals. The shared understanding requires an agreement about how the entire team will work together in an Agile manner. When projects require a large technical team, a key to success is finding a way to decompose the project into subprojects. Each subproject is then staffed by a Scrum team and managed by a Scrum of Scrums, which enables communication across projects and removes roadblocks that occur between teams.

For successful adoption of Scrum, it is critical to determine how these roles will be filled within the organization. The most common roles that are impacted in organizations during the change to Scrum are Product Manager, Program Manager, and Project Manager. Many organizations must significantly modify or remove some of the existing project or organizational roles to embrace and adopt these new Scrum roles. A key to success is transforming the roles so that a Scrum team is created that includes all the people necessary to deliver the product and only those necessary. Another key is to provide the team with clear goals and the freedom to decide how it will fill the roles and what the roles specifically mean to team members. Enabling the team to decide how to transition to these new roles and clearly communicating the new roles, responsibilities, and authorities is critical during the transition.

5. Collaborate Across Disciplines

The emergence of a self-empowered team is a sign that Scrum has been successfully adopted. For this to be attainable in most organizations, the team must be composed of all the disciplines necessary to define, build, validate, and prepare the software for release. This requirement means that the Development Team needs to include individuals from all disciplines that the adopting organization includes on its current project teams. While Scrum does not include specific guidance on determining the membership within the Development Team, it is common to include individuals from groups such as quality assurance, testing, technical writing, architecture, and possibly regulatory oversight groups as appropriate to the organization.

Many organizations making the transition to Scrum find that these groups have historically been established as silos and the shift to a small cross-discipline team can be a significant change. As part of the adoption, it helps to agree upon how they will work together and what they can expect from each other during a Sprint Planning meeting. The consensus should include agreement on how team members will work together on a daily basis, how decisions will be made, what it means to be "done" with individual tasks, and what it means to be "done" with Product Backlog items and Sprint Backlog items. It also includes decisions about any standards that will be used and how those will be validated.

Team members frequently have varying degrees of experience with and knowledge of the methodology. Therefore, during the formation of a Scrum team, it is important for everyone to develop a shared understanding of Scrum and about how they will use it. People often have different interpretations of how Scrum works and different perspectives on how Scrum will be used. A specific example is the varying definitions people have of what it means to be done with a Sprint Backlog item. For some, this can mean that code has been built and checked into the revision control system; for others, the definition includes, along with unit testing being completed, that unit tests are added to an automated framework and user acceptance tests are documented and completed. Having discussions about what it means to be done with Sprint Backlog items helps the team understand all the activities that need to occur and it brings out the expectations of each discipline on the team.

When teams are not familiar with Scrum, it can be useful to conduct training on the methodology to ensure there is a shared understanding. Once a common baseline of understanding has been developed, team members can discuss how they plan to work together during Release and Sprint Planning and revise this plan during the daily stand-up meetings to ensure it is working well for them. Sprint Retrospectives are also a powerful tool for determining what worked well during the previous Sprint and for identifying changes the entire team wants to implement on the next Sprint.

6. Balance Perspectives

During the adoption of Scrum, it is important to determine how much a team can and needs to provide long-term visibility into the direction of the project. While some teams adopt Scrum and provide visibility into the expected functionality of the project or system using just the Product Backlog, many organizations that Construx has worked with needed to find a way to balance the agility provided by Scrum with the necessary level of long-range planning.

To balance long-range planning with responsiveness to change, Construx recommends that organizations use multi-level planning during their Scrum adoption process to align team activities and promote a consistent understanding of the near-, mid-, and long-term goals for the project. The levels of planning are outlined in the following diagram and discussed in more detail below.

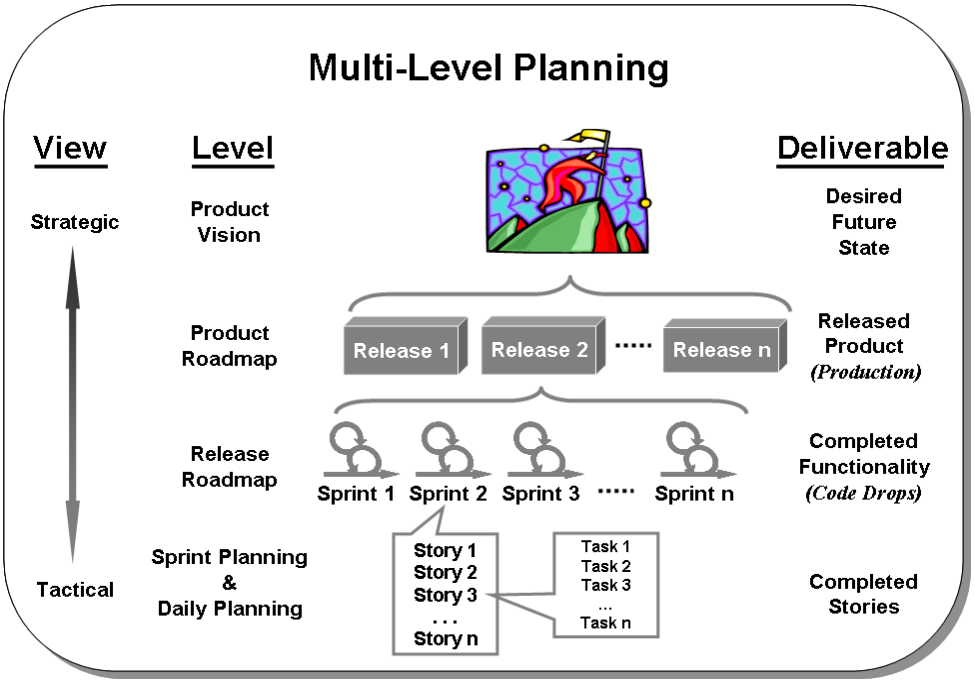

Figure 1 *Levels of planning.*

Use multiple levels of planning

The Product Vision communicates the current understanding of the desired end state. It describes the business objectives for the product, including the major objectives and relative priorities, and it outlines competitive positioning. It should provide clear guidance to all team members that support evaluating the functional and nonfunctional requirements to determine the subset of requirements that are most valuable in the near term. The vision should be reviewed and updated as the customer's needs change.

The Product Roadmap is the mid-level view that provides visibility across multiple releases. It describes the planned releases, marquee features for each release, technical strategy for the product, and target audience or customer base. It should include information about the direction of development for both the product and technology.

The Release Roadmap defines the overall strategy and a high-level path through the current release. It includes the prioritized goals for the release (for example, it specifies whether it is most important to automate existing functionality or provide new functionality to the users). It outlines the vision for the release and the strategy to achieve that vision, along with a description of each Sprint.

Appropriately balance predictability and agility

One key to success with this practice is to understand the stability of the system requirements. When using an Agile approach, the contents of the near-term releases are always more detailed and certain than the contents of the later releases. However, the

level of certainty can vary widely depending on the stability of the project scope and requirements.

When the requirements are fairly well understood and stable, the Product Roadmap defines the major user stories, outlines critical nonfunctional requirements, and defines the audience with a fair degree of confidence (e.g., 95% confident). The content of the next release outlines coarser grained stories (e.g., epics) with a lesser degree of confidence (e.g., 50% confident). The content of later releases describes only the major themes with a small degree of confidence (e.g., 25% confident).

The Release Roadmap reflects this level of certainty by having detailed Sprint visions and descriptions only for the earliest part of the project. Even with this limited detail, the team still reflects on the Release Roadmap at the completion of each Sprint to compare the progress to date with the expected progress and to ensure that business priorities and needs have not changed in light of new information or because of a better understanding of the system.

When the requirements are not well understood or the project is investigatory in nature, the Product Roadmap and Release Roadmap are very detailed only in the near term.

7. Invest in Essentials

The frequent releases of potentially releasable functionality require supporting essential infrastructure to ensure a high-quality product will be released. As teams plan the work for each Sprint, they need to include time to establish and maintain the infrastructure and practices that are critical for Agile development. This infrastructure includes having a solid build and test framework, creating and automatically executing unit tests, validating ongoing daily or continuous builds with an automated smoke test, and developing any standards or guidelines the team will use.

Establishing the infrastructure supports a system that is always integrated, deployable, and at a known level of quality. It reduces the risk that new features, changes to existing features, and updates to the underlying architecture will result in unexpected defects. It validates the quality of delivered functionality and improves the productivity of the team.

Incrementally build infrastructure

For new product development, the creation and maintenance of this infrastructure should begin during the first Sprint and continue for the duration of product development.

Organizations looking to adopt Scrum for an existing product often have more significant issues to deal with because the necessary infrastructure may not be in place. In some organizations that Construx has worked with, it has been necessary to put in place a daily build or continuous build process and a baseline of automated unit and system

testing before Scrum can be adopted. In other organizations, these items could be incrementally developed during the Sprints.

Create a plan to retrofit existing systems

Completely retrofitting existing systems with comprehensive unit and system tests is impractical. Instead, these should be incrementally added. Construx has seen numerous organizations start by requiring automated unit tests for new development, while selectively back-filling the areas that will benefit the most from automated regression tests. In support of this approach, we recommend that organizations profile their systems based on a set of criteria to identify the best areas to both put automated testing infrastructure in place and to back-fill tests.

Examples of aspects that could be used to profile the system for optimal automated unit testing areas include core parts of the system where reliability and stability is critical, modules that have a history of high defect rates or areas identified as being highly complex (by code-analysis tools, visual inspection, and/or general knowledge), and areas that are easier to test in an automated fashion because of either technological or environmental issues.

The infrastructure necessary to adopt Scrum will vary depending on the product, technologies, team, and infrastructure already in place. As an adopting organization, it is important to determine what is necessary to begin using Scrum and what can be incrementally developed during the Sprints. In either case, it is important to recognize that infrastructure investments are required and to include those items in the Sprint Backlog.

8. Steward the Architecture

A tenet of Scrum is that the requirements, architecture, and design of the system emerge throughout the project. However, without ongoing oversight and stewardship of the system architecture, an organization can end up with one of two undesirable results.

On one hand, a team can over-engineer the system in the early Sprints and spend time building an infrastructure or a design that is never needed in the system. On the other hand, a team can spend too little time on design and realize later in the project that the fundamental underpinnings of the system are too weak to support the objectives for the project. In the middle is a balance where teams consider the architecture and manage the level of technical debt throughout the project.

When adopting Scrum, an organization needs to ensure the team has one or more individuals who can guide the architecture of the product as it is being developed. In some more formal organizations, the team has an assigned architect who works with the team and with any external oversight committees to ensure the architecture is solid and fits within organizational or system architectural guidelines. In small teams or less formal

organizations, the team can simply meet during the Sprint to discuss architectural implications of the features in the Sprint and decide what needs to be done based on that.

One common approach that Construx has helped people implement is to use a risk-based approach for the architecture and design work throughout the project. When organizations have architectural risk or want to develop an overall approach, Construx often recommends that organizations conduct a Sprint 0 that shape the general architecture. The project team then incrementally builds upon that baseline and monitors how well the architecture continues to meet the products needs. After Sprint 0, the architectural and high-level design work will be done during each Sprint and the level of focus should vary depending on the risk and criticality of the features being implemented. When the team is concerned about the impact that a feature might have on the architecture, is working on a complex part of the system, or has identified technical debt that needs to be addressed, it might spend a larger portion of the Sprint on design work. To provide visibility of their efforts to stakeholders outside the team, the team often captures this work as a Product Backlog Item.

Construx recommends that throughout the system's lifetime the team incrementally capture the overall system architecture and high-level design description as it emerges throughout the Sprints. Unless required for external compliance reasons, this documentation does not generally include comprehensive details. Rather, it is a big-picture view of the major components of the system, along with the main interactions with other systems or business processes. The intent is to provide an overview of the system that is sufficient for enabling new engineers to quickly see the big picture. The medium for this information can be anything from a formal architecture document to an electronic form such as a Wiki.

9. Deliver Multiple Aspects of Value

One common mistake when adopting Scrum is to evaluate the features in the Product Backlog based solely on the end-user value they provide. This type of evaluation does not always provide a complete perspective of all the work necessary to release a product that will meet the users' needs and comply with all organizational constraints.

Construx recommends that the Product Owner and team take into account the following three elements when evaluating the value to be delivered in each Sprint:

- The customer value of having a specific feature in the system. This should include both a description of the functionality along with any nonfunctional requirements, such as how scalable, robust, or fast it must be.
- The technical value of building new infrastructure to support efficient software development, doing work now that is required to create functionality in future Sprints that is visible to end users, and refactoring an area of the system to pay down technical debt.

- The business value of doing work necessary to fulfill compliance requirements or meet the requirement of oversight processes, such as a Software Development Lifecycle.

Keeping all three of these aspects in mind when evaluating and prioritizing a Product Backlog helps the project team balance the need to provide ongoing value to the customer with work that is critical but not specifically related to product functionality. It can also result in work on the Product Backlog that is necessary to deliver value that cannot be directly linked to functionality visible to the end user that will be delivered in that Sprint.

10. Adapt with Purpose

Whether an organization is starting with an out-of-the-box deployment or a customized version of Scrum, the methodology usually will not exactly meet the needs of the organization and project during the first Sprint. The short duration of the increments provides a built-in opportunity to learn from experience and make needed changes. Most adoptions require at least a couple Sprints so that the organization can refine the deployment of Scrum to ensure it meets the organization's needs.

Some common areas that are refined during a Scrum adoption include:

- Changes in the definition of done for each Sprint
- Modifications to how the team works together
- Changes to team composition
- Introduction of additional software development processes and practices to support the Scrum methodology
- Refinements to the balance between responsiveness to change and long-term visibility

Once the initial adoption is complete, an organization needs to continuously learn from the experience and adapt the methodology more closely to its goals and practices. What works perfectly for the size of one organization, the complexity of its projects, and the external constraints it faces today might not meet the organizations needs in the future. Although it is uncommon to radically change how Scrum works within an organization after the initial adoption, it is common to incrementally modify Scrum as the organization changes.

Contributors

Jenny Stuart, VP Consulting

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

Earl Beede, Senior Fellow

earl.beede@construx.com
+1(425) 636-0114

Jerry Deville, Senior Fellow

jerry.deville@construx.com
+1(425) 636-0118

Eric Rimbey, Senior Fellow

eric.rimbey@construx.com
+1(425) 636-0109

Melvin Perez, Senior Fellow

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

About 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.

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

Construx®

SOFTWARE DEVELOPMENT BEST PRACTICES