Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictablity, and greater aptitude to respond to change. Scrum and Kanban are two of the most widely used Agile methodologies. Below are the most frequently asked questions around Agile and Scrum, answered by our experts.

WHAT IS AGILE?

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. Agile development refers to any development process that is aligned with the concepts of the Agile Manifesto. The Manifesto was developed by a group fourteen leading figures in the software industry, and reflects their experience of what approaches do and do not work for software development.

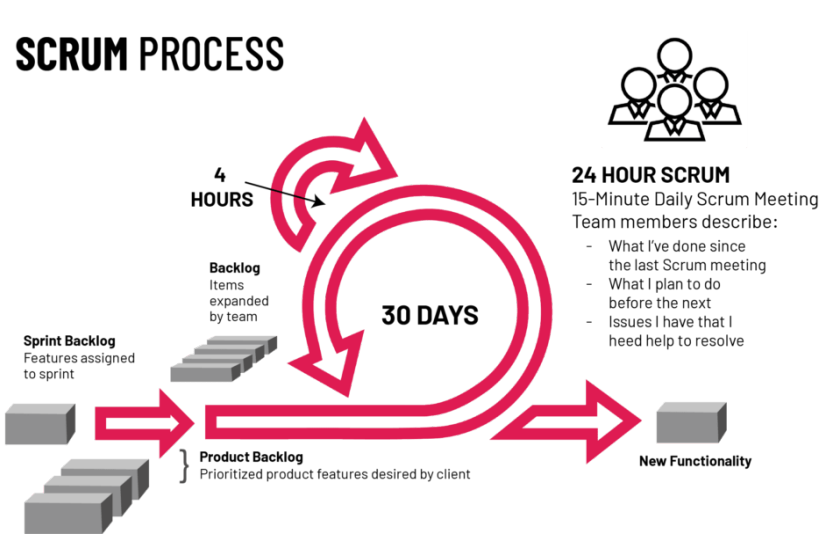
WHAT IS SCRUM?

Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one.

- A "process framework" is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints, the XP framework requires pair programming, and so forth.)
- "Lightweight" means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.

A Scrum process is distinguished from other agile processes by specific concepts and practices, divided into the three categories of Roles, Artifacts, and Time Boxes. These and other terms used in Scrum are defined below. Scrum is most often used to manage complex software and product development, using iterative and incremental practices. Scrum significantly increases productivity and reduces time to benefits relative to classic "waterfall" processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals. An agile Scrum process benefits the organization by helping it to

- Increase the quality of the deliverables
- Cope better with change (and expect the changes)
- Provide better estimates while spending less time creating them
- Be more in control of the project schedule and state



WHAT ARE THE BENEFITS OF AGILE?

Benefits to Customer

Customers find that the vendor is more responsive to development requests. High-value features are developed and delivered more quickly with short cycles, than with the longer cycles favored by classic "waterfall" processes.

Benefits to Vendors

Vendors reduce wastage by focusing development effort on high-value features, and reduce time-to-market relative to waterfall processes due to decreased overhead and increased efficiency. Improved customer satisfaction translates to better customer retention and more positive customer references.

Benefits to Development Teams

Team members enjoy development work, and like to see their work used and valued. Scrum benefits Team members by reducing non-productive work (e.g., writing specifications or other artifacts that no one uses), and giving them more time to do the work they enjoy. Team members also know their work is valued, because requirements are chosen to maximize value to customers.

Benefits to Product Managers

Product Managers, who typically fill the Product Owner role, are responsible for making customers happy by ensuring that development work is aligned with customer needs. Scrum makes this alignment easier by providing frequent opportunities to re-prioritize work, to ensure maximum delivery of value.

Benefits to Project Managers

Project Managers (and others) who fill the ScrumMaster role find that planning and tracking are easier and more concrete, compared to waterfall processes. The focus on task-level tracking, the use of Burndown Charts to display daily progress, and the Daily Scrum meetings, all together give the Project Manager tremendous awareness about the state of the project at all times. This awareness is key to monitoring the project, and to catching and addressing issues quickly.

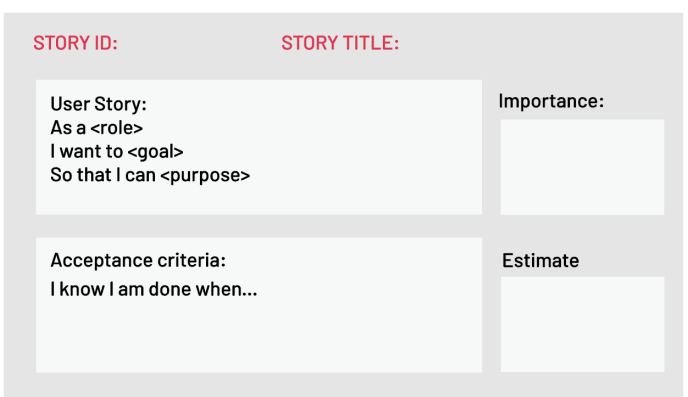
Benefits to PMOs and C-Level Executives

Scrum provides high visibility into the state of a development project, on a daily basis. External stakeholders, such as C-Level executives and personnel in the Project Management Office, can use this visibility to plan more effectively, and adjust their strategies based on more hard information and less speculation.

WHAT ARE THE SCRUM REQUIREMENTS?

Scrum does not define just what form requirements are to take, but simply says that they are gathered into the Product Backlog, and referred to generically as "Product Backlog Items," or "PBIs" for short. Given the time-boxed nature of a Sprint, we can also infer that each set should require significantly less time to implement than the duration of the Sprint. Most Scrum projects borrow the "XP" (Extreme Programming) practice of describing a feature request as a "User Story," although a minority uses the older concept of a "Use Case." We will go with the majority view here, and describe three reasonably-standard requirements artifacts found in Product Backlogs.

User Story



A User Story describes a desired feature (functional requirement) in narrative form. User Stories are usually written by the Product Owner, and are the Product Owner's responsibility. The format is not standardized, but typically has a name, some descriptive text, references to external documents (such as screen shots), and information about how the implementation will be tested. For example, a Story might resemble the following:

Name: Planner enters new contact into address book, so that one can contact the person later by postal or electronic mail

Description: Planner enters standard contact information (first and last name, two street address lines, city, state, zip / postal code, country, etc.) into contact-entry screen. One clicks "Save" to keep the data, and "Cancel" to discard data and return to previous screen.

How to test: Tester enters and saves the data, finds the name in the address book, and clicks on it. One sees a read-only view of the contact-entry screen, with all data previously entered.

The elements in this User Story are:

1. **Name:** The Name is a descriptive phrase or sentence. The example uses a basic "Role-Action-Reason" organization. Another common style, popularized by Mike Cohn, follows the template "As a <type of user>, I want <some goal> so that <some reason>." The choice of template is less important than having a workable standard of some kind.
2. **Description:** This is a high-level (low-detail) description of the need to be met. For functional (user-facing) requirements, the description is put in narrative form. For non-functional requirements, the description can be worded in any form that is easy to understand. In both cases, the key is that the level of detail is modest, because the fine details are worked out during the implementation phase, in discussions between team members, product owners, and anyone else who is involved. (This is one of the core concepts of Scrum: Requirements are specified at a level that allows rough estimation of the work required to implement them, not in detail.)
3. **Screens and External Documents:** If the Story requires user-interface changes (especially non-trivial ones), the Story should contain or link to a prototype of the changes. Any external documents required to implement the Story should also be listed.
4. **How to test:** The implementation of a Story is defined to be complete if, and only if, it passes all acceptance tests developed for it. This section provides a brief description of how the story will be tested. As for the feature itself, the description of testing methods is short, with the details to be worked out during implementation, but we need at least a summary to guide the estimation process.

There are two reasons for including the information about how to test the Story. The obvious reason is to guide development of test cases (acceptance tests) for the Story. The less-obvious, but important, reason, is that the Team will need this information in order to estimate how much work is required to implement the story (since test design and execution is part of the total work).

Story

Not all requirements for new development represent user-facing features, but do represent significant work that must be done. These requirements often, but not always, represent work that must be done to support user-facing features. We call these non-functional requirements "Technical Stories." Technical Stories have the same elements as User Stories, but need not be cast into narrative form if there is no benefit in doing so. Technical Stories are usually written by Team members, and are added to the Product Backlog. The Product Owner must be familiar with these Stories, and understand the dependencies between these and User Stories in order to rank (sequence) all Stories for implementation.

Defect

A Defect, or bug report, is a description of a failure of the product to behave in the expected fashion. Defects are stored in a bug-tracking system, which may or may not be physically the same system used to store the Product Backlog. If not, then someone (usually the Product Owner) must enter each Defect into the Product Backlog, for sequencing and scheduling.

WHAT ARE THE SCRUM ROLES?

The three roles defined in Scrum are the ScrumMaster, the Product Owner, and the Team (which consists of Team members). The people who fulfill these roles work together closely, on a daily basis, to ensure the smooth flow of information and the quick resolution of issues.

ScrumMaster

The ScrumMaster (sometimes written "Scrum Master," although the official term has no space after "Scrum") is the keeper of the process. The ScrumMaster is responsible for making the process run smoothly, for removing obstacles that impact productivity, and for organizing and facilitating the critical meetings. The ScrumMasters responsibilities include

- Teach the Product Owner how to maximize return on investment (ROI), and meet his/her objectives through Scrum.
- Improve the lives of the development Team by facilitating creativity and empowerment.
- Improve the productivity of the development Team in any way possible.
- Improve the engineering practices and tools so that each increment of functionality is potentially shippable.
- Keep information about the Team's progress up to date and visible to all parties.

In practical terms, the ScrumMaster needs to understand Scrum well enough to train and mentor the other roles, and educate and assist other stakeholders who are involved in the process. The ScrumMaster should maintain a constant awareness of the status of the project (its progress to date) relative to the expected progress, investigate and facilitate resolution of any roadblocks that hold back progress, and generally be flexible enough to identify and deal with any issues that arise, in any way that is required. The ScrumMaster must protect the Team from disturbance from other people by acting as the interface between the two. The ScrumMaster does not assign tasks to Team members, as task assignment is a Team responsibility. The ScrumMaster's general approach towards the Team is to encourage and facilitate their decision-making and problem-solving capabilities, so that they can work with increasing efficiency and decreasing need for supervision. The goal is to have a team that is not only empowered to make important decisions, but does so well and routinely.

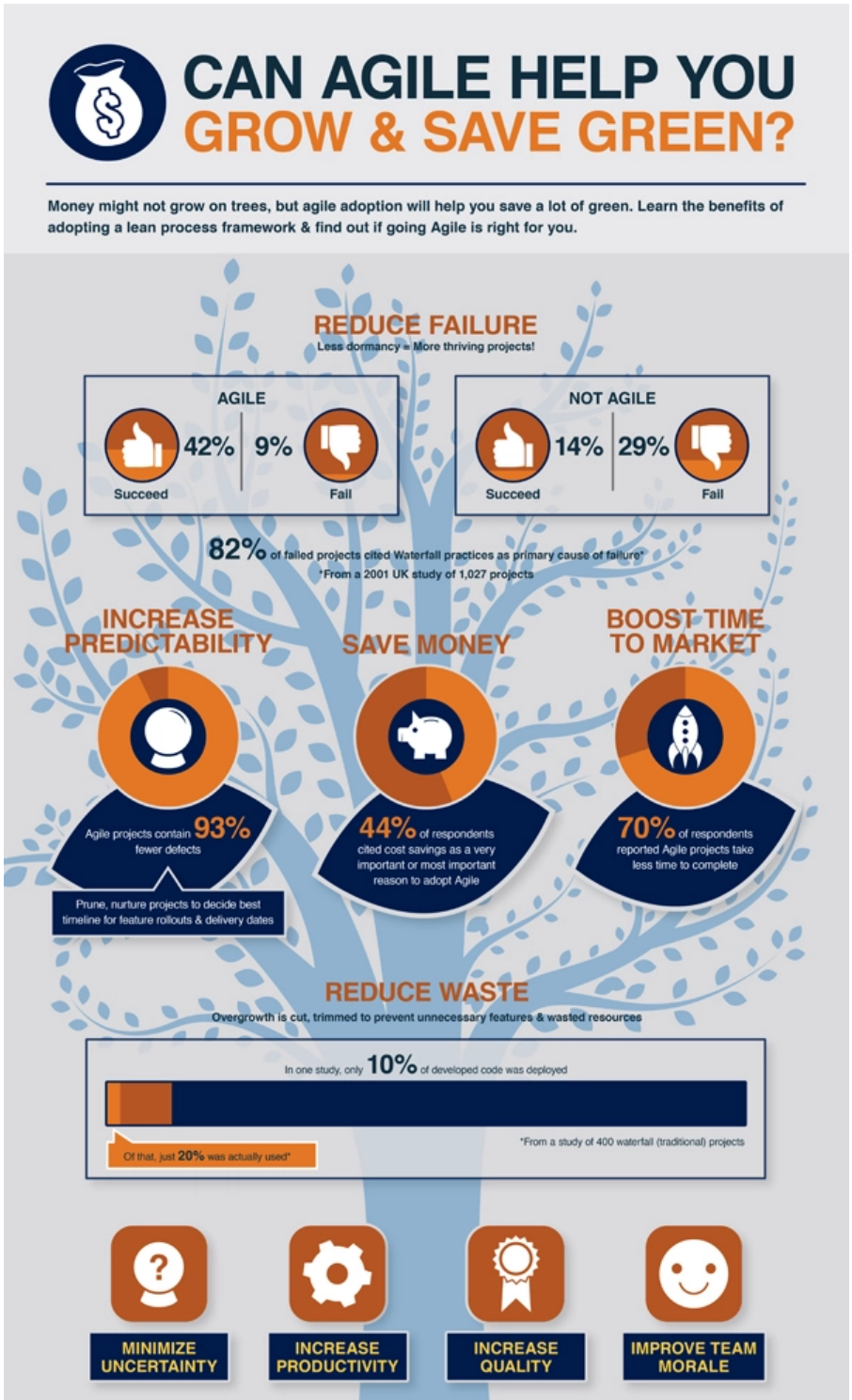
Product Owner

The Product Owner is the keeper of the requirements. The Product Owner provides the “single source of truth” for the Team regarding requirements and their planned order of implementation. In practice, the Product Owner is the interface between the business, the customers, and their product related needs on one side, and the Team on the other. The Product Owner buffers the Team from feature and bug-fix requests that come from many sources, and is the single point of contact for all questions about product requirements. Product Owner works closely with the team to define the user-facing and technical requirements, to document the requirements as needed, and to determine the order of their implementation. Product Owner maintains the Product Backlog (which is the repository for all of this information), keeping it up to date and at the level of detail and quality the Team requires. The Product Owner also sets the schedule for releasing completed work to customers, and makes the final call as to whether implementations have the features and quality required for release.

Team

The Team is a self-organizing and cross-functional group of people who do the hands-on work of developing and testing the product. Since the Team is responsible for producing the product, it must also have the authority to make decisions about how to perform the work. The Team is therefore self-organizing: Team members decide how to break work into tasks, and how to allocate tasks to individuals, throughout the Sprint. The Team size should be kept in the range from five to nine people, if possible. (A larger number make communication difficult, while a smaller number leads to low productivity and fragility.) Note: A very similar term, “Scrum Team,” refers to the Team plus the ScrumMaster and Product Owner.

HOW DOES AGILE SAVE YOU MONEY?



WHAT ARE SOME AGILE METRICS I CAN USE FOR REPORTING?

What to measure in agile is the enduring question. There should be two primary filters we should ask ourselves before we measure anything; "***will this measurement accelerate value delivery?***," and "***will this measurement enhance trust?***".

Below is an example of a fitting agile measurement. Typically an organization will create a goal to increase story point velocity, and this seems rational because we always strive to deliver more where possible. This perspective is looking at the problem from the wrong angle because what we want is value delivery not higher output. Those are not the same; one is outcome focused, and the other is production focused. Agile stresses outcome; which is value delivery, not output. Looking through the lens that equates increases in velocity to output assumes a few things; the teams are not working hard enough, and that output equals value. Of which both assumptions are typically untrue.

We should be using velocity to run our business; a story point velocity can be used to divide the product backlog and plan roughly when specific features will be available for our customers. What we need to do is incent stability in velocity, not velocity that is changing or in flux. In a world where there are incentives for increasing velocity, the teams will oblige and provide a higher story point velocity. They will inflate the story points to achieve the desired increase, which in turn reduce our ability to run the business because the velocity is no longer meaningful.

Addressed in a slightly different way we could measure the say/do of the sprint. Evaluating a team's estimate of how many story points they will deliver against what they perform in a sprint. Immediately the incentive causes stability in story point velocity, which provides the ability for the business to predict when features will release to market.

The former tells the teams they are not trusted, and erodes the creation of value delivery where the latter promotes both. It also catches teams doing good as their estimates start to converge to performance, moving to say/do of zero gives the business the ability to take the velocity to the bank as well as build trust within the organization. Everybody wins; our customers, our organization, and the team.

Sample Operational Metrics

- Lead Time
- Cycle Time
- Burndown Charts

Sample Output Metrics

- Throughput
- Agility Assessment Model
- Technical quality / defect measurements / code coverage
- # of features, etc.

Sample Outcome / Value Metrics

- Team Morale
- Customer Satisfaction / NPS
- Business Value

HOW DO I DEAL WITH DISTRIBUTED TEAMS IN AGILE?

There are two dimensions to this question. Having teams with remote team members and having all local teams where the teams are intact but in different geographical locations. Avoid the former at all costs.

Intact teams in different geographical locations
As with all problems, context is a primary constraint to solving this predicament. Companies that embrace these organizational attributes achieve best results; trust, and pulling the decisions to the place where the information exists. The people doing the work have the information; therefore this is a circumstance that should be left for the teams to solve themselves. The organization needs to trust, fund and support ideas coming from the teams regarding this difficulty.

The organization needs to support experimentation to all problem solving because that takes failure out of the conversation. Experiments require a known state, the desired state, and activities that move toward the desired state. Allow the teams to experiment, evaluate, and adjust to the new found learning resulting from that experience. Then be prepared to support a different approach and another experiment.

Having teams with remote team members
Having distant team members sucks for everyone. The communication disturbance is substantially worse which causes lack of awareness related to building products.

The soundest way to deal with this is to create all teams with local people. Challenge your thinking, assumptions, and constraints if colocated teams are not possible. As a last resort following the path described above should make the best of a terrible situation.

WHAT IS SAFE?

Agile development at the team or small organization level has emerged over the last 20 years as a really powerful way to improve delivery, engagement, and quality. Successfully and repeatably Scaling agile to medium and large organizations has been a problem, though. The Scaled Agile Framework (SAFe) has emerged as the leading solution to that problem. SAFe is a collection of principles, structures, and practices that has been shown to consistently and successfully scale Agile practices and deliver the benefits of Agile to organizations that had been working in waterfall or ad-hoc methodologies.

HOW DOES AGILE RELATE TO DEVOPS?

While Agile and DevOps started as independent methodological movements, they share a number of traits focused on improving the efficiency and speed of teams. As organizations become more Agile and refine their project management skill sets, they increasingly depend on technical teams being able to keep pace and maintain a certain flexibility.

This is where DevOps comes in as the "yang" to Agile's "yin". The DevOps approach helps development groups utilize new tools, automation, and different cultural strategies to change not just how they work themselves, but how they work with others. It becomes a symbiotic relationship where product teams work hand in hand with developers and testers and the like to ensure everyone has more contextual awareness. This promotes a greater overall quality of deliverables in a shorter period of time.

SHOULD I BE USING SCRUM, KANBAN OR ANOTHER FLAVOR OF AGILE?

Scrum is the dominant team based flavor of agile used today, it is over twenty years old and is time-tested. That said Kanban has its origins in manufacturing and Toyota applied it in 1953, another long-lived approach. Then there are various flavors of scaling frameworks to consider if organizational size is one of your contexts.

Context is primary to the answer. What commercial needs challenge your business? What size is your organization? How is your company organized? Are your core product teams dispersed in many geographical locations? Therefore the actual commercial problems your business faces and the way you respond to your customers are contextual to the answer.

Asking the question, "Scrum, Kanban or another agile flavor" is the first step and an excellent place to start. Considering a shift toward an agile approach is the first step toward sustainability. As described above agile is a requirement for future success, it is not new. Those organizations that do not adopt some form of agile will not be responsive to customer and market needs and are significantly disadvantaged.

HOW DO I SCALE AGILE ADOPTION?

Scaling agile is one of the most challenging issues to solve because there are so many variants of how organizations are structured and their commercial needs are diverse. This awareness brings into focus the notion of context.

Because of that diverse variance, there have emerged many scaling frameworks, and the notion of, "one-size fits all" is a false premise. Scrum is the dominant team framework; therefore, most scaling frameworks have Scrum at their core. Using Scrum as the basis to solve scaling problems is sound because most of them add to extend as a technique. As an example, the SAFe scaling frameworks introduce Kanban to facilitate the scaling challenges while keeping Scrum at its core.

The critical issues to consider when scaling beyond the team dynamic are; coordination, communication, shared or dependent work, and remoteness of groups or team members. These limitations are the same constraints at the team implementation of Scrum; however, as teams increase in numbers, they become amplified and extremely more difficult to solve. As an organization moves from one-team to multi-teams structure, broader issues become apparent. They tend to be the roadmap and investment rations between competing initiatives to support the vision and goals of the business.

Organization size also plays into the implementation and adoption of the scaling efforts as well as the scaling framework selected. A business of three hundred employees and an organization of tens of thousands employees require different approaches. Again illustrating the "one-size fits all" idiom.

To be sure the organizational scaling of Scrum is a whole company activity, not something isolated to product management and engineering as often occurs with Scrum implementations.

WHAT IS THE BEST HOLISTIC APPROACH TO AGILE ADOPTION?

Often when an organization adopts agile, the focus is on the engineering services group with some marginal collaboration with the product management department. This pattern is pervasive and typically explains why businesses do not feel that they receive the benefits they expect from an agile adoption. Furthermore the conjecture that agile does not work.

Commercial needs, company size, organizational structure, and a host of other considerations create the context needed to frame an approach to agile adoption. By far the leading success system requires the inclusion of all aspects of the business. System thinking, that of understanding that all domains of the company accomplish value delivery are aligned and working together. Therefore to ask the engineering department with some support from the product management department become agile misses the mark.

Unfortunately, the business will more than likely have to consider restructuring and shifting management styles to achieve organizational alignment. Best outcomes happen when the leadership team goes all in with an open mind to the possibilities when they collaborate. Collaborate with a focus on value delivery and working in a supportive way recognizing that they all will reshape in support of those possibilities.

Some examples are when the accounting department transition from Cost Accounting to Lean Accounting. Human resources department considers the moving to OKRs and eliminating MBOs and KPIs. The company metrics focus on measurements that correlate to value delivery over output.

The best approach when considering an agile adoption relates to your organization's context as described above. Then be inclusive with the leadership team and their areas of focus with an eye to accelerated value delivery over output and utilization.

HOW DO I AMPLIFY THE IMPACT OF AGILE?

By developing a learning organization with the benefit of a clear purpose and providing an environment where people are trusted.

Learning Organization
What does being a learning organization mean or imply? The fundamental principals of Scrum are inspect, adapt, and transparency. Embedded in the Scrum principles and are present in every event as feedback loops. They intended to have as many learning opportunities as possible and experienced as frequently as possible.

We need to involve the entire company in these principles because the higher benefits from agile are dependent on system thinking. System thinking coupled with the focus on value delivery. We desire the measurements that influence the engineering services to be consistent with what drives the business.

Clear Purpose
The organization needs to provide a purpose that is bigger than the individuals within the organization, a goal that is larger than the organization itself. It needs to touch at the emotional level of everyone, and it should be the inspirational reason people want to come to work.

Trusting Environment
The goal is to have the ability for everyone to experiment and learn. Experimentation is different than merely failing at something. If we construct an experiment, a few things must be defined. A current state, the desired state, and the experiment itself that moves toward the desired state.

Given that set of constraints, the experiment yields either a supported outcome or a null hypothesis. And both are significant data points that should influence our future behavior.

In conclusion, agile is a company-wide sport, and it is not merely an engineering services activity. Without all three; learning organization, clear purpose, and trusting environment, the effects of agile will be diminished.

HOW HAVE OTHER ORGANIZATIONS SUCCESSFULLY ADOPTED AGILE?

Context is essential, the framework for a change as dramatic as altering the way a company operates requires leading the staff through the journey as opposed to dragging them. Some critical areas for success are to recognize that change is difficult, and an acknowledgment that this endeavor is a human effort.

Embracing the Scrum values of commitment, courage, focus, openness, and respects. And expressing them in ways that the company entirely embraces them, so they become organizationally shared values will promote success.

A dynamic approach to seeking volunteers will surface staff looking for positive change and filter out those opposed to change. This strategy will remove the organizational blockers from the transition because they are not part of the progress toward the new operational method. As time progresses the change begins to have visible outcomes; happier staff, innovation grows more pronounced, and value delivery becomes accelerated. Suddenly there becomes momentum as staff, teams, departments, and business units become pulled toward the new operating model of agile.

The issue of scaling agile is monolithic therefore starting at the team, or a few teams are the beginning of the journey which is required. Caution against applying scaling frameworks on day one typically yield less than beneficial results in the long run.