Seven Questions to Ask to Determine if Your Organization is Agile Ready
Elizabeth Larson, PMP, CBAP, CSM, Co-Principal, Watermark Learning, Inc.
Richard Larson, PMP, CBAP, Co-Principal, Watermark Learning, Inc.

Abstract

These days, organizations want the ability to respond rapidly to an ever-changing business environment. Some organizations have heard that adopting Agile methods provides quick project turnaround. Not all organizations, however, know how to make Agile projects successful. When considering the adoption of an Agile framework such as scrum, some wonder how to apply it to “real life.” If your organization wants to take advantage of Agile but is having difficulty applying the principles, you may be wondering whether your company is ready for Agile. How can you be sure? The answer is by taking a seven-question organizational readiness self-assessment to see if Agile/scrum is right for your organization. The questions on the self-assessment can be used to begin conversations, so that important issues are raised and resolved, ensuring a successful conversion to Agile.

Specifically, this paper:

• Describes why organizations need agility
• Describes the common mistakes made in Agile adoption
• Discusses why by managing expectations Agile is more than the latest buzzword for software development
• Explores why some organizations succeed in adopting Agile methods while others do not
• Provides a simple seven-question self-assessment to determine if your organization is ready to adopt an Agile framework
• Provides tips for discussing the results of the assessment with the real decision makers

What is Agile?

Agility is defined as “nimbleness” and the “power to move quickly and easily” (dictionary.com). It has sometimes been equated with speed, so that an Agile project has been misinterpreted to mean one that is completed quickly. The opposite of agility, though, is clumsiness or stiffness, not slowness. Organizations want to be nimble and flexible in order to adapt to business change. They want new products and services without waiting for months or even years, and they want to be involved, to collaborate, to make business decisions, and to know that the final product will work when it’s delivered to them. Agile development addresses these needs.

Agile, as applied to projects, is a set of methods based on principles that allow us to move through our projects quickly and easily without a lot of bureaucracy. It is comprised of lightweight practices and principles, rather than processes and templates. Agile methods are not a methodology, which is prescriptive and tells us how to do our work. Scrum, although often called a methodology, is a framework rather than a “full-blown methodology, [and as such] scrum is deliberately incomplete” (Martin, 2010, ¶1). That is, scrum is a skeletal structure designed to support the Agile methods.

One of the significant benefits of Agile is that it helps manage expectations through:

• Collaboration, which is a cornerstone of the Agile methods. The business and the development team work together, ideally co-located, which encourages the creation of strong relationships built on mutual respect and trust, which in turn usually increase productivity and reduce development time.
• Transparency, which is not only important in managing expectations, but also in building trust. One way Agile methods encourage transparency is by having the team’s work displayed prominently on a wall dedicated to the team’s project, so that everyone—the CEO, IT management, the sponsor, and the product owner—can see the team’s plan and progress. The team wall includes all the requirements, usually written as user stories, showing which have been completed, which are in progress, as well as those promised, but not yet started. It also displays all the activities that need to be completed and the associated estimates. Gone are the days of the “traffic light” dashboard status reports, showing all projects as “green,” even when some are in trouble.
• **Quality.** The business establishes the quality criteria, so their expectations of how well the product works and whether it can be released are set early in the iteration (known in scrum as a *sprint*).

Agile in general and scrum, in particular, manage expectations, because business experts:

• Write the requirements. The business representative on the project, known as product owners (or customer), writes high-level requirements in the forms of user stories, which are simple narratives describing the desired feature or function. Having the product owner write these simple narratives helps ensure that the focus is on the business and not technology. The collection of these user stories comprises what is known as the *product backlog*.

• Prioritize the requirements and all changes based on business value. They are the ones who determine how important each feature is and where it belongs on the prioritized list of requirements (product backlog).

• Facilitate reviews of the product, helping to ensure that feedback is received from other affected business stakeholders. Requested changes are added as user stories to the product backlog and re-prioritized.

• Participate in lessons learned sessions, which are called *retrospectives*, to recommend ways to improve the development process.

**Agile Manifesto**

There are four articles in the Agile Manifesto (Agile Alliance, 2001), and they are often misinterpreted by organizations as they begin to adopt Agile methods. Each article starts with the words “We value” and each has two phrases separated by the word “over.” The phrase on the left is valued “over” or *more* than the one on the right. Although the latter is not discouraged, it is valued less than the phrase on the left. Let’s look at each article, how it has been misinterpreted, and the problem each is trying to solve.

**Article One. We value individuals and interactions over processes and tools.**

This article is sometimes misinterpreted to suggest that everyone on the project can do whatever he or she wants, as long as the job gets done. The problem this article is trying to solve is being hampered by a bureaucratic, cumbersome one-size-fits-all process applied to all projects and using a requirements management tool, even when the complexity of such tools is not warranted. Common sense and the appropriate processes and tools should certainly be used, and many Agile teams have benefited by using consistent processes and tools, such as automated testing tools.

**Article Two. We value working software over comprehensive documentation.**

This article has been misinterpreted to mean that there is no need for documentation on Agile projects. The problem being solved is the necessity of writing a large requirement specification just because the process requires it. Some projects require more documentation, such as when there are regulations and legal or compliance requirements, when the project is global, the team cannot be co-located and team members speak different languages, when the project has many subcomponents that need integration and those subcomponents will be built by different people in different locations. The preceding examples all require more documentation than those for small, self-contained projects. We need to apply the right amount of documentation for our projects.

**Article Three. We value customer collaboration over contract negotiations.**

I doubt if too many organizations would forego contract negotiations. However, many projects fail because the vendor and the performing organization have argued over the contract, specifically relating to whether a change was in or out of scope. Agile methods promote collaboration between vendor and the customer, which is essential if we are going to meet the project objectives.

**Article Four. We value responding to change over following a plan.**

This article has been misinterpreted to mean that planning is not necessary on an Agile project. The problem that needs to be solved is the practice of baselining the scope without allowing for changes or with a change process so cumbersome that it doesn’t seem worthwhile. Agile addresses the need to provide an easy way to handle changes through its time boxes and reprioritization process.
Agile Principles

Supporting the Agile Manifesto are twelve Agile principles (Agile Alliance, 2001). Below I have categorized these principles into two groups: customer focus and quality. Note that many of them are good project management principles that have been in place on traditional projects for many years. Agile, however, formalizes these principles.

Customer Focus

1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** Early delivery has been misinterpreted by some to mean quick. Early delivery means taking an iterative approach. This iterative approach is one of the three phase-to-phase relationships cited in *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* (PMI, 2008, p. 22). Although some traditional projects are completed iteratively, most are done sequentially, with one phase completing before the other begins. For example, the entire business analysis phase is done before signoff can occur and before design can begin, which means that often final products are not delivered for months or years. This Agile principle reflects the need to produce something of value sooner rather than later.

2. **Deliver software frequently from a couple of weeks to a couple of months, with a preference for the shorter timescale.** Breaking large deliverables into smaller, more manageable pieces has always been a best practice in project management. This Agile principle emphasizes its importance. Imagine the advantage to the organization of getting new features and functions in a matter of weeks, rather than having to wait many months or years.

3. **Welcome change even late in development. Agile processes harness change for the customer’s competitive advantage.** Agile is known as a change-driven approach (IIBA, 2009, P. 18). We not only need to expect change, we need to welcome it. Rarely do products that are designed with the original requirements produce happy customers. Welcoming change means putting scope change decisions in the hands of the customers who decide whether or not the changes belong in the project.

4. **Business people and developers must work together daily throughout the project.** “Working together” requires collaboration and implies that both the business and the technical staff are equally important to the success of the project. It implies that both groups are available and have access to each other. Gone are the days when subject matter experts (SMEs) were unavailable for requirements definition, yet were expecting the project delivered by the “promised” date. The business as represented by the product owner is accountable for the requirements, but working with the delivery team each day helps ensure that they are actually implemented.

5. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** Note that this principle does not say that it’s the only way to communicate, but rather that it’s the most effective. The farther away team members are from each other, the longer communications will take and the greater the risk of miscommunication.

Quality

1. **Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.** When team members are motivated and feel empowered, they tend to produce quality results. A supportive environment built around trust and respect usually creates an atmosphere that not only motivates most team members, but also encourages pride of ownership.

2. **The best architectures, requirements, and designs emerge from self-organizing teams.** Self-organizing teams see what needs to be done and take the initiative to get the work done. There are no specified roles, and tasks are not assigned. Self-organizing and self-directed teams take tasks from the list of outstanding work. There is no assigned team leader, but leaders tend to emerge, regardless of their title or role.

3. **Working software is the primary measure of progress.** The best way to judge the quality of a product is whether or not it works as expected, not the length of a requirements specification or the adherence to organizational standards. When we deliver a product that doesn’t work or is not usable, we cannot call it a quality product nor can the project be considered a success. Although there are often other metrics used to judge the performance of both the product and the project, the quality of the end result is the key metric.
4. **Continuous attention to technical excellence and good design enhances agility.** Agility does not promote “quick and dirty” development. Instead, spending enough time on technical excellence and good design helps ensure a quality product.

5. **Simplicity—the art of maximizing the work not done—is essential.** Agile methods stress the importance of prioritizing all work, including changes, based on business value. The expectation of the entire team is that unless there is business value, the work will not get done. Therefore, work that on many traditional projects was completed without question will not be completed on an Agile project, because it doesn’t make business sense to do so. Maximizing the work not done helps minimize scope creep.

6. **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.** Agile project teams that have some history of working together can calculate their productivity rate, which means how much work can be completed in each iteration. This rate is known as velocity. In addition, the actual work accomplished is compared each day to where the team expects to be. This information is placed on charts called burn-down charts. Because Agile teams become really good at estimating how much they can accomplish in each iteration, heroic efforts, with lots of overtime at the end of the project, are not necessary, and a constant rate can be maintained.

7. **At regular intervals, the team reflects on how to become more effective, and then fine-tunes and adjusts its behavior accordingly.** As with all quality processes, we need to continually improve our processes in order to produce better products. In Agile, we complete our lessons learned, often called retrospectives, to determine how we can improve, and the team determines the adjustments needed, which are put in place for the next iteration

**Absorbing Change**

A major benefit of Agile is breaking large projects into very small pieces and implementing tiny features and functions in short intervals; however, each new feature or function introduced into production means a change in at least one business process. In addition, each time anything is released into the production environment, even if very small, it increases the risk that the production environment will not work. Absorbing change every few weeks may not be possible for all organizations and/or divisions or business areas within the organization. Here are some considerations:

- Business changes. Can the organization change its business processes every few weeks?
- Technical changes. Do your production change control processes allow for frequent changes?
- The organizational culture can greatly influence the ability to absorb change. Because Agile methods introduce a great deal of change, organizations that “have always done it this way” may not easily adopt Agile. Organizations whose culture encourages trying new things and learning from them may find Agile adoption easier.

Some organizations adopt Agile, but do not follow all the practices. Some scrum practitioners call these “scrumbuts”; some examples of “scrumbut” include:

- We’re going to introduce scrum, but we don’t want to have product reviews.
- We’re going to implement scrum, but we’re not going to do release planning.

There are often compelling reasons why organizations choose to adopt some practices but not others, so we really need to be sensitive to the organization’s culture.

**Seven Questions to Determine if Your Organization is Agile Ready**

What follows are seven questions that we can ask to help determine if your organization is really ready to adopt Agile and they are based on common adoption mistakes we have seen in our practice. The questionnaire focuses on the organization’s commitment to devoting the necessary time and resources to make Agile work. In other words, it is aimed at determining an organization’s commitment to adopting Agile.

1. **Is your organization willing to dedicate a full-time business expert, called a product owner?**

   This is the role that represents the business community, particularly the project sponsor (sometimes called business owner, the main customer, or the stakeholder), and therefore is tasked with answering the business questions and making business decisions. This is the go-to person for the requirements, for answers to the team's questions about
the features and functions that the business wants and how the business will use those features and functions. This is also the role that prioritizes the requirements on the product backlog.

Product owners need the wherewithal to be effective. The team needs to be able to count on them to make business decisions quickly. To make those decisions, they need to understand the business, which includes the business processes, the information used, and the interaction of the end-user with the end product. They need to understand it so well that they can make decisions without having to go back to other business stakeholders during the iteration to get answers to the team’s questions. They also need the positional authority to make those decisions. There is not enough time in an iteration to have the sponsor or other business stakeholders second guess the product owner’s decisions. Finally, they need to be available. In the middle of the sprint, the team needs immediate access to the person making the business decisions. In order to complete the sprint in a short time frame, such as two weeks, the team cannot wait for the product owner to get back to them at his or her convenience. They cannot wait for the product owner to check with other subject matter experts (SMEs) to come to a consensus. The team needs immediate answers. Having a dedicated product owner is critical to the success of an Agile project.

2. Is your organization willing to dedicate a full-time delivery team?

Many organizations require that their project managers, business analysts, and team members work on multiple projects, and they are reluctant to supply the dedicated resources for Agile projects. They question why resources need to be dedicated on Agile teams. There are many advantages to having a dedicated team, for example:

- Productivity is lost when resources have to stop working on one project and start working on another.
- It is harder to calculate the productivity rate (velocity). Because productivity is lost moving among multiple projects, it is more difficult to know how much can be completed in an iteration/sprint.
- Working on multiple projects, especially when there are different team members on each project, means adjusting to the team dynamics on multiple projects. The team members need to learn how to work through the inevitable disagreements that arise. They need to determine how the team will make decisions. They need to understand each other’s strengths and weaknesses, all of which take longer when team members work on multiple projects.

Self-organizing teams tend to be high-achieving teams; nevertheless, producing new features or functions every few weeks, no matter how effective the team is can feel like intense pressure and concentration is needed. In an interview (NPR Staff, 2011 February) about his book on high achievement, Clutch, Paul Sullivan addresses five ingredients that lead to success, saying that these characteristics pertain to successful athletes, business people, lawyers, and even parents. The five ingredients are:

1. Focus
2. Discipline
3. Adaptability
4. Being present
5. Pushing away from fear and the pull of desire to do well

Team members who work on multiple projects usually find it difficult to focus. They also find it hard to “be present” when struggling to solve problems related to other projects or figuring out how to work with members of other teams.

3. Is your organization willing to provide a business analyst to elicit just-in-time (JIT) requirements?

Just-in-time does not mean jumping straight to design, as often happens when the team tries to define requirements during the iteration. It means spending the time to “groom the product backlog,” which involves defining detailed requirements before they’re needed for the current sprint. JIT requirements means that requirements are completely elicited before the beginning current sprint, and completely defined means that they have been translated so that a product can be designed, built, tested, and implemented. One clear advantage is that when requirements are "groomed" or defined before each sprint, time is not wasted during the sprint trying to figure out what each user story (requirement) means. The product owner does not need to take time away from the delivery team to meet separately with other SMEs to uncover needs and expectations. That work has already been completed. It also means
that the data requirements are understood, the requirements support business processes, and any prototypes have been developed and approved. Finally, it means that a detailed definition of “done” has been completed, often in the form of acceptance criteria. Grooming the product backlog is a perfect role for the business analyst.

To understand the importance of drilling down requirements, we will provide a user story example. As noted earlier, user stories are written as simple textual statements and kept at a high level. Each user story has the role of the end-user, what he or she wants to accomplish (goal) and the benefit (motivation)—all from the end-user perspective. For example,

As a customer service representative (role), I need to be able to search for customer information (goal) in order to respond quickly to complaints (motivation).

This user story is high-level. Just a few of the things we need to determine are what is a customer, what kind of customer information is required, how does that information relate to other information needed by this role, what are the business rules surrounding this information, and what, if any, are the permitted values. We also need to understand the process of responding to complaints, not to mention the need to define what “quickly” means. However, if we wait until the current iteration to answer these questions, we won’t have enough time during the iteration to design, build, and test the product.

4. **Is your organization willing to time box each iteration?**

A time box is a fixed duration into which an agreed amount of work is completed. In Agile, once a team has worked together long enough to understand its velocity, the duration of each time box is established. The team estimates how many user stories can be completed in that time box, given its resource capacity. Time box durations do not change.

Agile teams often ask whether or not it is important to time box each iteration because, they explain, in their companies the “powers that be” keep adding features to the iteration, which extends the number of days. In these organizations, the time boxes become fluid and it’s hard to plan what can get done during the iteration. Time boxes are important on Agile projects for a number of reasons:

- Time boxes help set expectations. The business knows what to expect from each time box. For example, if each iteration is two weeks, the business knows that every two weeks it will be getting a new feature(s).
- They are a useful planning tool. The team can estimate how much scope (how many user stories) will be completed during each iteration. This estimate becomes a commitment from the team to the business of what will be completed.
- Time boxes help control scope and reduce the risk of scope creep. No new features are admitted into the time box. If changes are identified, the product owner writes them as new user stories, prioritizes them, and adds them to the product backlog.

5. **Is your organization willing to put the right people in the right roles?**

The roles on a scrum project include:

- Delivery team, without titles or role delineations. Each member of the delivery team determines what needs to be done, how long it will take, and does it. In theory, this team includes business analysts and testers, but successful Agile teams have worked with separate business analysts and testers.
- The product owner, as described earlier, represents the business and has the time and authority to make business decisions relating to the product. In that role, he or she is accountable for creating, prioritizing, and communicating requirements and requirements status to the rest of the business.
- Scrum masters are the chief roadblock (impediment) removers. In addition, they plan the resource capacity, facilitate status meetings (called the daily scrum), measure performance to plan (burn down and burn up), and protect the scrum processes.
Some organizations do not want to allocate resources to these separate roles, so they put the wrong people in the right roles. Examples include:

- Business analyst as product owner. *A Guide to the Business Analysis Body of Knowledge (BABOK® Guide)*, Version 2.0 states that the business analyst is a practitioner who will “recommend solutions that help an organization achieve its goals.” (IIBA, 2009, p. 3). Business analysts do not make business decisions and have no authority to do so.

- Business analyst as scrum master. Having a business analyst in the role of the scrum master is similar to having the business analyst in the role of the project manager. The scrum master is responsible for the project, and the business analyst assists the product owner with the product. The disparate interests and skills required support keeping the roles separate.

- Technical architect as scrum master. The latter is a neutral facilitator, completing such tasks as removing roadblocks, planning the resource capacity, and monitoring the status with the burndown chart. The technical architect needs to make technical decisions, so this role confusion is not usually helpful to the Agile team.

Companies often put the wrong people in the right roles either because they do not understand the importance of these roles or they are not willing to make a complete commitment to the Agile methods. In either case, Agile adoption is apt to be more difficult for these companies.

6. **Is your organization willing to support a collaborative environment?**

   Remember the old saying “Information is power?” (Robin Morgan, nd, ¶12) Collaboration is based on the axiom that sharing information is power. An organization that is siloed and hierarchical in its reporting structure and communications style is not one in which trust is typically engendered or where the organization feels a need to respond quickly to change. Organizational silos create barriers that separate “us” from “them,” so communications take longer, and the risk of a distrustful atmosphere is high. Organizations have learned that in order to get things done quickly and productively, they have to share information and collaborate. Generally speaking, the more siloed the organization, the less chance of successful Agile adoption.

   In his article on the advantages of a collaborative over a siloed organizational culture, Evan Rosen (Rosen, 5 February 2010) describes some of the negative impacts of a siloed organization and the benefits of collaboration, as well as the roadblocks an organization can expect as it transitions between the two.

   Creating a collaborative environment can build trust, create an environment in which the team feels empowered, and can ultimately reduce the time it takes to communicate and produce the product features in each iteration. Cultures that do not foster collaboration cannot be truly successful in adopting Agile practices.

7. **Is your organization willing to apply the necessary discipline?**

   Although scrum practices are lightweight, they are less effective when applied randomly. Some teams, although they call themselves “scrum” teams, are what are sometimes referred to as “cowboy developers.” Organizations that are looking for a “quick and dirty” way to develop software are probably not looking to implement Agile methods in their organizations. To be adopted effectively, the scrum framework, even though it is based on a lightweight set of practices, requires a certain amount of rigor.

   The amount of rigor on an Agile project is based on the type of project. For example, a project being completed for a heavily regulated industry may require more rigor than one without any legal or compliance requirements. However, regardless of the type of project, we still need to initiate, plan, execute, monitor and control, and close the project and iterations as follows:

   **Initiating.** We still need something like a Project Charter that authorizes the project; it should include the business need, business and project objectives, and some high-level description of the end-product or service. Many Agile projects use a vision statement to provide direction to the team as a way to initiate the project.

   **Planning.** We plan releases, sprints, resource capacity, how much work will be completed during the iteration and what will be done in the next 24 hours.

   **Executing.** We execute each sprint. Each day we complete what was planned the previous day.
**Monitoring and controlling.** Each day we report on our progress and measure our plan against our actual work, using burn-down and/or burn-up charts. These visual representations of where we planned to be and where we are help guide the team and set expectations with the product owner. Completing reviews of the product during sprint demos is a way to find and prioritize product defects.

**Closing.** Retrospectives, or lessons learned, help formalize process improvement.

Exhibit 1 presents the seven factors for successful Agile adoption.

<table>
<thead>
<tr>
<th>Factors for Successfully Adopting Agile</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-time product owner</td>
<td>Needs the authority, ability, and availability to make business decisions</td>
</tr>
<tr>
<td>Dedicated delivery team</td>
<td>Need to be focused and “present”</td>
</tr>
<tr>
<td>Just-in-time requirements</td>
<td>Need a balance between getting all requirements up front and waiting until the iteration to elicit the detail.</td>
</tr>
<tr>
<td>Time-boxing the iteration</td>
<td>Fixing the duration and calculating how many user stories will fit into the iteration is an effective scope management technique.</td>
</tr>
<tr>
<td>Having the right people in the right roles</td>
<td>Too often organizations put the wrong people in the right roles due to lack of understanding and/or commitment.</td>
</tr>
<tr>
<td>A collaborative environment</td>
<td>The greater the collaboration, the more successful adoption of Agile is apt to be. Siloed organizations and departments tend to struggle with Agile implementation.</td>
</tr>
<tr>
<td>Applying the necessary discipline</td>
<td>The Agile practices and principles, although lightweight, require discipline to produce quality results.</td>
</tr>
</tbody>
</table>

**Exhibit 1: The seven factors needed for successful Agile adoption**

**Difficult Conversations**

If after taking this seven-question self-assessment we find that our organization is not ready for Agile, how can we proceed? We need to understand what business problem adopting Agile will solve. Is this someone’s solution in search of a problem? Here are some situations and a possible direction for starting this difficult situation.

- The organization wants to speed up development time and has heard that Agile development is quicker than traditional development. We need to educate these types of organizations about what Agile is and what it is not. We need to discuss the discipline and commitment required. Finally, we need to make sure that we understand the underlying problem that is causing the need for a quicker response and fix that problem, either by adopting Agile or in some other way.
- The IT organization wants to adopt Agile because it has heard about the positive results on Agile projects. Again, it is important to find out what about Agile appeals to the IT staff and why they think that adopting Agile would benefit the whole organization. If the IT area is ready for Agile but the rest of the organization is not, we need to point out the risks that could arise from lack of commitment, availability, or collaboration, which could result in a failed attempt to implement Agile.
- The organization wants to use Agile as an excuse to speed up development time, even if products are defective and the environment is chaotic. We need to point out the risk of introducing (or continuing) “cowboy” development and the impact on the business and staff. We need to lay out the Agile processes, prerequisites, and requirements before them and recommend the appropriate thing for the organization.

**Summary**
There are many advantages to an organization that adopts Agile methods, including happy customers, motivated teams, and quality products. Organizations that have put the necessary resources and practices into place have seen major gains in productivity; however, those organizations who want to use Agile as an excuse to produce defective products quickly or who do not have the necessary commitment, may be disappointed in the results.

References


February 6, 2011 by NPR Staff


This material has been reproduced with the permission of the copyright owner. Unauthorized reproduction of this material is strictly prohibited. For permission to reproduce this material, please contact PMI or any listed author.