[FAQs on Agile](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile)

[Skip to end of metadata](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#page-metadata-end)

* Created by [Silverman, Michael](https://www.csa.sungard.com/wiki/display/~Michael.Silverman), last modified by [Kadam, Deepak](https://www.csa.sungard.com/wiki/display/~Deepak.Kadam) on [Aug 15, 2014](https://www.csa.sungard.com/wiki/pages/diffpagesbyversion.action?pageId=61900159&selectedPageVersions=6&selectedPageVersions=7)

[Go to start of metadata](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#page-metadata-start)

* [What is Scrum?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhatisScrum?)
* [How is it different?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Howisitdifferent?)
* [Why is it better?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Whyisitbetter?)
* [Is it easy to learn?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Isiteasytolearn?)
* [Will Scrum solve all of our problems?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WillScrumsolveallofourproblems?)
* [It sounds stressful and monotonous – is it?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Itsoundsstressfulandmonotonous–isit?)
* [What is Lean and what is Agile?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhatisLeanandwhatisAgile?)
* [I heard that Lean was invented by Toyota, is that correct?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-IheardthatLeanwasinventedbyToyota,isthatcorrect?)
* [Who first thought of using Lean principles to build software?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhofirstthoughtofusingLeanprinciplestobuildsoftware?)
* [Are there other Agile software development methodologies?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-ArethereotherAgilesoftwaredevelopmentmethodologies?)
* [I have heard of something called Kanban, what is that?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-IhaveheardofsomethingcalledKanban,whatisthat?)
* [What is difference in Doing vs. Being Agile?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhatisdifferenceinDoingvs.BeingAgile?)
* [That sounds good, can we use that instead of Scrum](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Thatsoundsgood,canweusethatinsteadofScrum)
* [I am not in development, why do I need to care?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Iamnotindevelopment,whydoIneedtocare?)
* [What are the considerations before starting a Large-Scale Agile Adoption?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhataretheconsiderationsbeforestartingaLarge-ScaleAgileAdoption?)
* [Who is in charge of Agile development projects? The CIO? The project manager? The Scrum Master?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhoisinchargeofAgiledevelopmentprojects?TheCIO?Theprojectmanager?TheScrumMaster?)
* [What is a Scrum Master's role? Is the Scrum Master more like a project manager than an Agile leader is?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhatisaScrumMaster'srole?IstheScrumMastermorelikeaprojectmanagerthananAgileleaderis?)
* [What is the role of the C-level executive in Agile development projects?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-WhatistheroleoftheC-levelexecutiveinAgiledevelopmentprojects?)
* [What is a product owner?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Whatisaproductowner?)
* [Do we really need a product owner?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Dowereallyneedaproductowner?)
* [How/when you get your velocity?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-How/whenyougetyourvelocity?)
* [How do you increase velocity?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Howdoyouincreasevelocity?)
* [What constitutes a release?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-Whatconstitutesarelease?)
* [What's the best way to account for maintenance and other non-project work?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-What'sthebestwaytoaccountformaintenanceandothernon-projectwork?)
* [What's the purpose of Iteration 0?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-What'sthepurposeofIteration0?)
* [What's the purpose of the MOU?](https://www.csa.sungard.com/wiki/display/agile/FAQs+on+Agile#FAQsonAgile-What'sthepurposeoftheMOU?)

**What is Scrum?**

Scrum is a software development process framework that helps implement a number of lean and agile principles, such as continuous improvements, incremental development and delivery, collaborative team and people oriented work, quality focus, and rapid time to market.

Using a small team, meeting every morning in the daily stand-up meeting to coordinate and optimize the execution, Scrum lets you turn customer needs into working software in three weeks. This is done by using an ordered list of features that are implemented one by one, continuously testing, checking quality, and documenting. We encourage face-to-face conversions, multiple people collaborating on the same features, and learning.

In a nutshell:

* Split your organization into small, cross-functional, self-organizing teams.
* Split your work into an ordered or prioritized list of small, estimated,concrete deliverables. Assign someone to be responsible for this list.
* Split time into short fixed-length iterations (1-4 weeks).
* Split releases by having a synchronized, steady release cadence, quarterly or monthly.
* Optimize the process by having a retrospective after each iteration.

For more information, see <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100> or <https://www.scrum.org/Resources>.

**How is it different?**

In essence it is feed-back driven, not plan driven. This means that it uses empirical measures to adjust continuously and to better predict outcomes. Working software is the primary measure of progress, as opposed to milestones or project reports.

**Why is it better?**

Maybe the question should be, is it better?

To answer that it may be useful to look at some statistics. Below is the graph showing the Chaos report that studies projects in terms of on time, on budget delivery. Even looking at that narrow measure of success Agile seems to outperform traditional forms of project methodologies. If we add to that the fact that the incremental feed-back driven possibilities in Agile allows the customer to get what they actually want by pivoting and letting the customer change their minds, the difference in customer delight is probably even greater.

**Is it easy to learn?**

Yes. But like chess it is easy to learn but can take a lifetime to master. We constantly need to learn and improve ourselves. We also need to challenge and improve the process, and our standards. If we do not, we can rest assured that the competition races on.

**Will Scrum solve all of our problems?**

Scrum exposes problems. It does not fix them. It is designed to quickly show impediments so that we can act and remove them. Scrum does not fix a company or a system. The fixing needs to be done by the team members, management and the company.

Examples:

* Your testing is too slow to have time for verification needed to have potentially releasable software every month.
* You can’t make the necessary priority decisions between a number of items that do not fit in the time available.
* Testing and documentation cannot be done with the same speed as you develop features.
* Customers cannot take new versions to get the benefit of improvements because of incompatibility or upgrade problems.
* Development capacity in one area of the product is too low.

In each case, one should not try to avoid the problem, letting it fester, instead the root cause needs to be identified, and a lasting remedy should be implemented. At least the first steps towards a remedy should be achieved.

**It sounds stressful and monotonous – is it?**

[The time boxes and continuous flow are there to protect the teams. Work should be limited to capacity, and the pace should be such that it can be kept indefinitely.](http://www.mountaingoatsoftware.com/uploads/blog/Agile-Waterfall-Success-Failure-Rates.jpg)

**What is Lean and what is Agile?**

A truly Lean organization is one where there is a system of massively parallel processes that react to customer and market demand to develop and produce exactly what they want, when they want it – all while continuously seeking perfection to be able to react faster, with better quality and less waste. This is in contrast to old style centralized command-and-control organizations using large batch production trying to anticipate demand and holding large inventories.

Agile is an umbrella term for software development methodologies focusing on small complete teams building the smallest possible useful part of a product, give it to users, who tell you what is right and what is wrong. Incrementally your solution will home in on the target and hit it even if it is moving.

Lean management principles can be used to effectively scale agile development across multiple teams and products, and making it repeatable.

See the Agile manifesto for more on Agile: <http://www.agilemanifesto.org/> or <http://www.lean.org/whatslean/> to read more on lean.

**I heard that Lean was invented by Toyota, is that correct?**

The Toyota Production System or the Toyota way is a major precursor of the more generic Lean.

**Who first thought of using Lean principles to build software?**

The term lean software development originated in a book by the same name, written by Mary Poppendieck and Tom Poppendieck. See <http://www.poppendieck.com/>   

**Are there other Agile software development methodologies?**

Yes, for example eXtreme Programming, Crystal Clear, DSDM. Scrum is today by far the most widely used one, but it is worth noting that many of the practices from eXtreme Programming fit very well inside Scrum, and some are almost mandatory. See http://www.extremeprogramming.org/

**I have heard of something called Kanban, what is that?**

According to Wikipedia,**Kanban** (かんばん(看板)) (literally signboard or billboard) is a scheduling system for lean and just-in-time (JIT) production. Kanban is a system to control the logistical chain from a production point of view, and is not an inventory control system. Kanban was developed by Taiichi Ohno, at Toyota, to find a system to improve and maintain a high level of production. Kanban is one method through which JIT is achieved.

[Kanban in a nutshell:](http://www.mountaingoatsoftware.com/uploads/blog/Agile-Waterfall-Success-Failure-Rates.jpg)

* Visualize the workflow
* Split the work into pieces, write each item on a card and put on the wall
* Use named columns to illustrate where each item is in the workflow.
* Limit WIP (work in progress) – assign explicit limits to how many items may be in progress at each state.
* Measure the lead time (average time to complete one item), optimize the process to minimize this.

In software development, Kanban or sometimes ScrumBan is used when the requirments change on a daily basis, or there is a random inflow of task, like in a helpdesk or emergency room. While Scrum requires the team to deliver production quality software at the end of each iteration, Kanban or ScrumBan requires almost continuous production quality.

**What is difference in Doing vs. Being Agile?**

Many teams believe they are Agile because they are following the prescribed Agile practices. E.g. A team may think they are Agile because they are doing daily stand up meetings, sprint planning meetings, retrospectives etc.

The mistake that teams often make is that they tend to treat these practices as ‘ceremonies’, thereby focusing more on getting the ‘ceremony’ done, and not so much on doing it right so as to derive the benefit that the practice is intended to provide. Take, for instance, daily stand up meetings. A team can be said to ‘be’ Agile if they can achieve the following benefits through these meetings:

* Instilling a clear sense of purpose about what needs to get done
* Focus on efficiently moving the work forward
* Early identification of risks and blockers
* Team members helping each other with share obstacles

However, many teams think they are Agile, just because they are ‘doing' stand up meetings, even when their stand up meetings are not delivering any of these benefits, or worse, have anti patterns.

Doing Agile practices does not translate into being Agile, unless there is focused effort put in by all team members to derive the intended benefits from that practice. Teams need to be conscious about why they are doing a practice and should aim to keep improving the maturity level of that practice.

To state in other words, treating Agile as a noun is futile. Teams are Agile only when they understand that Agile is an adjective and constantly work towards it and also adopt the Agile mindset. The mindset change is required to make the transition from just ‘doing’ Agile to ‘being’ Agile.

**That sounds good, can we use that instead of Scrum**

If your special circumstances require even shorter term plans than normal iterations, it can be considered. Note that normally some cadence is used anyway (nested or synchronized), for demos, retrospectives, predictability etc.

**I am not in development, why do I need to care?**

Agile and especially Lean are management principles, not just a development process that can be used by all parts of the company; client services, professional services, and even sales. Important parts of this management principles are servant leadership and self-organizing teams. Continuous, iterative, team-based improvements can significantly increase a software company’s overall performance.

Lean is a philosophy that organizations have adopted to develop the agility needed to meet overwhelming global challenges — reducing waste, enhancing iteration speed, and continuous improvement through improved innovation.

These principles are:

1. Add Value to the Customer
2. Eliminate Waste
3. Create Knowledge
4. Respect People
5. Build Integrity In
6. Defer Commitment
7. Deliver Fast
8. Optimize the Whole

These principles are nested in a mindset of creating an organization that continuously improves itself and its process, while respecting its people with a focus on adding value to its customers::

* A relentless pursuit of eliminating waste
* Adding as much value to your customers as quickly as possible
* Creating and managing knowledge
* Respecting and growing your people.

**What are the considerations before starting a Large-Scale Agile Adoption?**

***Management***

**1. Do you have Management’s support?**

Among the challenges a team may face in the early stages of adopting Agile, the most important one is winning the confidence and support of top management. If you do not have a genuine commitment to change from your top executives, you are not likely to be successful. Win their confidence support before you set out your journey. Be upfront with them on the roadblocks you may face and lay out your plans on how you intend to overcome them.

In some cases, the senior executives and junior technical staff may be enthusiastic to “go Agile,” but there may be a lot of skeptics in middle management. This could be due to the fear of losing control over their teams or inability to distinguish Agile development from “cowboy coding” or apprehension on the quality of the product or lack of documentation. You need to allay their fears by showing concrete examples.

What if your leadership fails to agree with you? You can still go ahead with Agile, but you are on your own in dealing with the challenges.

**2. Is Proper Training Provided to the Organization?**

The industry-standard training for Scrum teams is the Certified ScrumMaster (CSM) training course, conducted by a Scrum Alliance Certified Scrum Trainer, and we strongly recommend that this training be provided not only to ScrumMasters but also to team members, managers, and others in the organization that will be interacting with the Scrum teams. In addition, it's imperative that Product Owners complete Certified Scrum Product Owner training. If great results are to be achieved, everyone needs to understand how to “play the game” well.

This training in the fundamentals of Agile is just the foundation for success, though, and to be successful, there are other new skills that will have to be learned. First, Agile development quickly makes visible any deficiencies in teams ability to work together closely and produce high quality functionality in the span of an iteration. Many teams require training to learn better coding and testing practices, whether in the basics of test automation, refactoring techniques, or in practices such as Test-Driven Development. In many organizations, team members have highly specialized skill sets, and often have difficulty working closely together as part of a cross-functional team; they may require coaching in how to do this successfully. Often, the most effective way to retrain the team in cross-functional behaviors is to “seed” the team with one or more members who are experienced in this way of working.

Another good way of accelerating learning is to start a study group that meets periodically to discuss various aspects of Agile development before the training takes place. During this period the team can read literature available on the Internet, starting with the Scrum Alliance website [[www.scrumalliance.org](http://www.scrumalliance.org/)]. We found these meetings are extremely helpful to come onboard quickly in terms of understanding the new practices and mindset, and adopt it to our environment.

In addition to new technical skills, teams may also require more education in their product domain. In Agile, developers are not “software robots,” blindly implementing software specifications; rather, we want them to be full partners in the development effort, helping the Product Owner achieve maximum business value. For this to occur, they need to have a working understanding of the business and technology context for the work they are doing.

Many organizations make a solid investment in training in Agile technical practices, but overlook training in the soft skills needed to really get the most from those practices. ScrumMasters, Product Owners, functional managers, and executives should receive training in facilitation skills, Agile retrospective techniques, dispute resolution, root cause analysis, and systems thinking. All of these are practical necessities for skillfully coping with the myriad dysfunctions that Agile development will quickly surface.

**3. Did you train your Product Owners?**

Strong product ownership is a key to the success of Agile development, since it is this group that bridges the gap between the end customers, management and IT. A very thorough understanding of the responsibilities of the Product Owner, how to work with customers, the team and stakeholders to achieve maximum ROI, and how to be effective with the essential practices are required.

When faced with large-scale development effort, it is often recommended that all teams work from a single, project-wide Product Backlog, as opposed to having multiple, team-specific Product Backlogs; this will help reduce the duplication, redundancies, and conflicting directions that can often result from the latter.

Unless your Product Owners are equipped with these skills, you are not going to see the big wins from Agile adoption.

**4. Do you have enough space?**

Agile principles emphasize face-to-face communication wherever possible. If the individual members of your Agile team sit at different locations, the overhead of communication reduces their effectiveness. It is mandatory to rearrange their seating so that it improves informal communication within the Scrum teams. Get the teams co-located into their individual team spaces even if you have to convince your facilities, HR, VP or CEO.

You also need to ensure that they have enough free space for informal gathering around a computer for a quick walk-through, their daily stand-up meetings etc. Also, ensure that the teams have adequate wall space, white boards, flip charts and writing material.

**5. Are all your support teams on board?**

Organizations make all efforts to train the teams that are adopting Scrum. But, often they ignore the need to familiarize other support teams. If the development team is going to be successful with Scrum, they need to have full support of any specialists who provide support but are not members of the team, for example, DBAs, Sys Admins, or the Configuration Management team. Unless they know the constraints and limitations of your Scrum teams, you are not likely to receive their deliverables and meet your Sprint commitments.

Get at least one representative from each of these departments enrolled in the Scrum training. Try to get these specialists included in your Scrum teams, ideally as full-time members of the team, or at least in the role of consultant.

**6. How quickly to scale?**

Often, large organizations attempt to roll out Scrum across several teams simultaneously; what some people call the “Big Bang” approach to change. Tempting though this may be, the risk of such an approach is that systemic or large-scale dysfunctions (which afflict many teams across the organization) will be surfaced by many teams simultaneously, and will have to be resolved quickly and for many teams at once. This is both very challenging and very chaotic. It may be easier to begin Scrum with a small number of teams, try different solutions to solving these baseline dysfunctions, and be able to provide best practices to later teams that follow in their adoption of Scrum. As the organization gains confidence and competence in Scrum, more teams can start practicing it.

**7. Is an appropriate attrition strategy in place?**

This may not be an important item in difficult economic times, but for organizations susceptible to high attrition, dealing with this problem could be very challenging. Unlike in traditional models, Scrum believes in tightly gelled teams of individuals with a core competency and multiple cross-functional skills. As Scrum teams are smaller, losing a single member could prove to be a significant setback for a team. Even if you find a quick, equivalent replacement, it will take some time for the new hire to get up to speed, bond with the team and start delivering.

Scrum provides the benefit of teams, but also makes more visible the disruption when teams are broken or changed.

Have a trained pool of engineers available, who could substitute an unexpected departure in your team.

**8. Can your engineers appreciate the purpose behind a user story?**

A competent technical team can convert a user story into database tables, Controllers, tags and Form elements fairly quickly. But some teams may have difficulties in comprehending the motivation or ROI behind a feature they are implementing. For Agile to be successful, it is critical that all members of the development team (developers, testers, etc.) “get into the Product OwnerΓÇƒs head” and appreciate the rationale behind a user story. They should keep questioning the objective of major decisions and suggest alternative implementations where possible.

Otherwise, one risks ending up with brilliant technical implementations that fail to meet the business goals.

**9. Module assignment to teams**

If you are planning to assign different software modules to different Scrum teams and you expect them to have exclusive check in privileges on their modules, you need to be very cautious. While it is a good idea to let individual Scrum Teams master specific software components, if you implement a user story that requires multiple Scrum teams to work on it, all hell breaks loose. The interface design, hand off, and integration between multiple teams introduce a lot of waste.

Also, avoid restricting teams from touching other modules based on their ownership. You can better enforce the code ownership with good code reviews and refactoring policies.

***Code***

**10. How tightly coupled is your system?**

If your code base consists of tightly coupled modules, and you have a large number of Scrum teams, chances are that they will be stepping on each othersΓÇƒ toes constantly. The problem will be more acute if the Sprint or release cadences of the teams donΓÇƒt align.

If possible, spend some time re-organizing your code base to minimize the inter-module dependencies. This exercise is well worth the effort as the code will be more modular and less tightly coupled, which in turn will create better „codeΓÇƒ conditions for the Scrum Teams to do their job, and enable the teams to more quickly produce more business value.

**11. Do you have good Unit/Automation test coverage?**

At the heart of Scrum is the delivery of potentially shippable software at the end of each Sprint. For this to happen, test automation is a must. If you do not have a comprehensive set of automated unit tests that are tied to your build system, you are likely to spend more time in testing and debugging than necessary, and it will be difficult to end each Sprint with a fully tested (and fixed) system. In addition, automated tests give the team confidence in making changes; they know that if the change breaks other parts of the code, it will be immediately visible.

Strengthen the foundations of your code base by adding as many unit tests and automation tests as possible. Teach your developers the art of Refactoring, if they don't already know it. Tools like Emma, Jester and Cobertura will help you measure the coverage of unit tests.

***Tools***

**12. Do you have good collaboration tools?**

This item is critical for the geographically distributed teams. Agile requires a close interaction among team members, so you need to ensure your team has all it needs for effective communication. Use video conferencing, telephones, emails, instant messaging, and Skype as much as possible. VNC, WebEx, Sococo are other useful tools for the team collaboration. If you are planning to introduce a tool such a Rally, VersionOne, or ScrumWorks, make sure the teams themselves have an opportunity to evaluate them fully before making a decision.

**13. Do you have a source code repository replication system?**

This item is more relevant for multi-location teams. Nothing can be more frustrating for your remote teams than accessing the repository on a slower network. Due to shorter release cycles, your teams would be doing more frequent checkins, checkouts, branching, merging than they normally would if they were using traditional methodologies. Make sure you have a fast, reliable repository replication solution in place before you roll out Scrum.

**14. Is your build and deployment system fully automated?**

It is difficult to imagine a modern software development organization not wanting to automate their build and deployment activities. Design your build and deployment processes to be completed in a short time, say, 30 minutes. This should include executing the test cases and certifying the build. It is even more preferred to have a continuous integration (CI) policy before you move to Scrum. The CI will highlight the build/integration issues and help the teams identify and fix the issues quickly. When the teams are working in short time-boxed iterations, any time gained in their iterations will be valuable.

Alternately, if you require a lot of manual intervention in the build and deployment, your teamsΓÇƒ velocity is going to be affected.

***Others***

**15. Do you have adequate Dev and Test environments?**

If you would like to have dedicated Dev and QA environments for individual (or groups of) Scrum Teams, you need to plan them in advance. You also need to ensure you have enough hardware, software licenses, and IT support staff to build or maintain those environments. Since budgeting, getting approvals, ordering, hiring, and training take a while, you need to plan this well in advance.

**16. Do you have enough monitoring/health-check systems in place?**

If you are working with a large number of environments and each of these environments has multiple servers, databases, networks and other third party components, you need to have a robust monitoring system in place. Zabbix and Hobbit are a couple of good tools you could consider for monitoring the health of your systems.

For more inputs on Agile readiness visit <http://blog.sei.cmu.edu/post.cfm/organization-ready-for-agile-fourth-336>

**Who is in charge of Agile development projects? The CIO? The project manager? The Scrum Master?**

The short answer is no one individual, and everyone on the team. That's because the tenets of the Agile Manifesto call for self-directed, self-organizing teams to work together on projects. In this model, team members handle specific tasks suited to their ability and the project's needs, and the team is collectively accountable for the project's results. Agile proponents believe that spreading out management tasks empowers all players and spurs innovation and productivity more than old-style command-and-control management does.

The long answer is that most Agile projects include members who take on Agile management roles and become team leaders. These team leaders come in many forms: the Agile coach, whose role is to help the team understand and use Agile practices; the basic Agile team leader, who is a facilitator of meetings and consensus; and the Scrum Master, who in Agile-like development is considered closest to a project manager. To most Agile proponents, as long as collaborative development takes place, the titles of Agile team leaders don't matter.

On the face of it, the role of the traditional project manager is eliminated in Agile development. In Agile circles, traditional project management is considered too top-down, hierarchically speaking. In contrast, Agile favors collaboration, continuous self-improvement and responsibility sharing. Management tasks are shared by all members of the development project team.

That's not to say that Agile development is anti-project manager. Its aim is to relieve project managers of having to take responsibility for the whole project -- and taking the fall for failures. Agile parses out some, but not all, of the project manager's responsibilities across the whole team. For example, assigning daily development-tasksand decision making go to the development representative; quality management has a representative on the team, while quality assurance is a shared responsibility. In this situation, project managers don't make the decisions that should reside with experts in each area, and that should result in faster delivery and better product quality.

Whether or not a C-level executive starts the Agile ball rolling, it's critical that senior management support Agile practices and participate in projects. Agile, however, doesn't take away the need for project management skills. Instead, it calls for a change in style -- from someone being "boss" to being "coach" and "team member" -- and sharing responsibility and evolving as iterative development practices change.

**What is a Scrum Master's role? Is the Scrum Master more like a project manager than an Agile leader is?**

The leader of a Scrum development project is called the Scrum Master. That person facilitates the processes of the team, just as an Agile team leader does, but also ensures that the Scrum process is followed.

Scrum is a development methodology that is usually placed in the Agile camp because Scrum and Agile have similar practices, such as frequent, short team meetings and iterative development. Some Scrum purists disagree, and some Agile evangelists consider Agile a cultural methodology.

When you think about the role of the Scrum Master or Agile team leader, keep the ideal of collaboration, collective decision making and responsibility-sharing in mind. In first-time Agile projects, for example, having a project manager or an Agile coach can help the team best use Agile processes to break away from the productivity impediments of Waterfall development. The Agile methodology encourages experimentation, so organizations can adapt to new leadership options as their Agile expertise and experience grows.

**What is the role of the C-level executive in Agile development projects?**

CEOs, chief information officers (CIOs) and chief technology officers usually make strategic decisions about Agile in large enterprises, ranging from initial adoption to resource planning to scaling Agile and more.

Frequently, the call to adopt Agile comes from upper management. Whether or not a C-level executive starts the Agile ball rolling, it's critical that senior management support Agile practices and participate in projects. Upper managers' key role is providing the resources and the leadership to make the organizational and cultural changes the Agile methodology requires. In most cases, senior managers are the ones with the authority to bring business, IT, development, customers and other constituents together as a team. Senior managers may also hold the purse strings for training and staffing.

Many Agile project failures and slowdowns can be attributed to lack of senior management involvement in Agile adoption, training and individual projects. CEOs, CIOs and senior managers might not have the time to attend all stand-up meetings, but their participation in Agile project teams encourages buy-in from non-development members of the team. In addition, senior managers can bring information about business goals and roadmaps that helps teams with building application requirements and estimating iterations. Their sponsorship at the project level is critical, Galen said, to the delivery of the right software at the right time.

**What is a product owner?**

The product owner is a new role, created and defined by the Scrum Alliance ([www.scrumalliance.org](http://www.scrumalliance.org/)). Product owners live full-time with development teams - elaborating users’ stories, managing sprint-level backlogs, expanding specifications, and interpreting product vision.

Most product companies already have staff members with similar skills, such as a requirements analyst or business analyst (titles and job descriptions have shifted over time, but product companies have always needed to provide developers with detailed, feature level information and UI guidance; someone with intimate customer experience is always necessary to build great software).

The product owner addresses the agile development teams’ intense need for real-time input on user stories, user experience/user interface, and requirements.

In Scrum, a single person must have final authority representing the customer’s interest in backlog prioritization and requirements questions. This person must be available to the team at any time, but especially during the sprint planning meeting and the sprint review meeting.

Challenges of being a product owner:

* Resisting the temptation to “manage” the team. The team may not self-organize in the way you would expect. This is especially challenging if some team members request your intervention with issues the team should sort out for itself.
* Resisting the temptation to add more important work after a Sprint is already in progress.
* Being willing to make hard choices during the sprint planning meeting.
* Balancing the interests of competing stakeholders.

**Do we really need a product owner?**

Many agile development teams think they need a product owner for every project, but product owners address only a small portion of the agile product management challenge. The product owner role has been created by the agile development community to focus on the hour-by-hour demands of an agile team—even though software product companies typically have a product manager and other existing resources to meet those needs.

A product manager’s focus is creating products that are delivered to a market full of customers, not one-time projects delivered to a single customer. And if we have agile product managers assigned to the product, do we ever need a product owner?

A thoughtful answer needs context, and must be based on the structure and talents of the team. Agile development is scaling beyond single co-located teams and now includes revenue-driven product organizations. The four situations that demand for a product owners attention are:

1. Single, co-located agile team
2. Large, co-located projects with multiple development teams
3. Geographically distributed development teams
4. My product manager isn’t doing a good job and doesn’t know agile development

**How/when you get your velocity?**

There are ways to calculate velocity for a new team that involves breaking down stories into tasks and providing time estimates for each task and then the team guessing at what they could get done in an iteration. But the best way is simply to have the team work for 2-3 iterations and then use a safety factor to calculate a velocity for release planning. If a team has already been working together, they should have an established velocity and methodology for estimating. (This is one reason to keep teams together.)

**How do you increase velocity?**

*Picture this scenario: Your boss wants your team to deliver certain functionality by a certain date (deadline), but your velocity is unable to achieve that. What options/suggestions do you have for your management who really want this to get done? More people? Overtime? What else will increase the velocity?*

First, do Scrum. The point of Scrum is to improve your ability to deliver. So tell us a bit about what Scrum is telling you ... or why it isn't telling you.

1. Your mission is to build shippable software in every sprint: **try to do that and pay attention.**
2. You have a daily standup meeting to tell you what is in people's way: **remove those obstacles.**
3. You have a retrospective every Sprint to decide what needs improvement: **do that.**

Second, to speed up you need to know what is slowing you down. There are many possibilities, and your actions will depends on the answer to following:

* are you delivering a shippable product every two weeks?
* is your product owner competent and empowered?
* are requirements / user stories clear?
* are you encountering & processing many defects?
* are you writing tests before the code, or after?
* are there many handoffs from person to person in the course of building a feature?
* are your features large and complex, requiring more than a couple of days to build each one?
* are team members available full time on the project?
* are there changes and interruptions causing work to be thrown away or redone?

In short, what's happening that is slowing you down? Quite likely, the team knows what to do about those things. However, with a deadline looming, you're not going to suddenly go twice as fast, add more people, work overtime.....a simplistic answer is to choose what to do next, and to find and defer work that's not priority.

**What constitutes a release?**

Ideally a release should be an organized into themes and contain a static number of iterations (~ 2 - 8 iterations is ideal). This makes the deploys predictable for testing and for clients. Also, teams can better focus when they are all working towards providing functionality for a smaller number of themes. There are times when release dates will need to be adjusted. This is fine as long as it is communicated to and agreed to by the project staff and clients.

Releasing to production is ideal because it provides value to the clients sooner. For some projects, the entire system needs to be completed before it goes to Production, so a release would just be to Test. At a minimum, deploying the application to a Test environment at the end of each release allows clients to test the application and provide feedback earlier in the project timeline.

Every team will develop it’s own cadence to iteration and release timelines. Each project will have its own constraints, so basically, do what makes sense for your project.

**What's the best way to account for maintenance and other non-project work?**

Teams seem to handle this one differently, so do what fits best.

The release plan should take into account possible production support, holidays, vacation, etc. The simplest way to do this is to reduce your velocity for planning purposes. Your team may be able to get 50 points of work done per iteration, but 10 points may be towards support. So, use 40 points for planning. You may also simply add an extra iteration into a release because of holidays (e.g. Christmas/New Year's) or if you know people are going on vacation.

Some teams have a Support line on their task board. This is so people can put up tasks for things they are working on that are outside of the main project. These are not estimated or counted in the team velocity. They are there so people know who is working on what. Again, during planning, we know that every team member has meetings, support work, and other duties that they need to perform. People probably only work on true project work for 4-6 hours of a work day.

If there is a known large support issue during a project, then this could be added to the plan. For instance, if you know your team may be involved in testing for the new desktop image, some features should be added for that, even though it's not directly related to your project.

And yes, all buffers should be built into the release plan and the Product Owners / clients should be told if there any uncertainties in the deploy dates.

**What's the purpose of Iteration 0?**

Any iteration (Iteration 0) is for what you need it to be to get your team ready to start developing features. This could include planning and most definitely will entail infrastructure work for new projects. Maybe the team needs to research some new tools that will be required for the project or simply wants to prototype some piece of functionality that will be risky. It is important to "timebox" Iteration 0. It should last no longer than 2 - 4 weeks. You should include Product Owners, Product Managers, Architects and whomever else that you need.

**What's the purpose of the MOU?**

The MOU serves to 1) provide goals for the project, 2) general scope and constraints, and 3) an opportunity to discuss issues that may affect the project (and other projects). It gets everyone on the same page and allows other teams to determine if this project may affect them.

Any uncertainty with the release dates should be explained during this time. If it's an enhancement project, the end date may be constrained by session so the clients should be informed that features may be cut if time runs out. For a rewrite, the end deploy date may have to be moved if more time is needed to complete the core features.

The scope should be talked about at as a high level as possible and may be broken down into "will do" and "as time permits". And if a release plan is still in the works, then again, this uncertainty in the deploy dates should be explained.