Adopting Agile for Large Companies
Table of Contents

Introduction/Summary

Start with the Pilot Project
  • Pick the Right Team
  • Team Environment
  • Size, Duration, and Scheduling
  • Organizational Support
  • Importance and Criticality
  • Minimal Risk
  • Avoid New Technologies
  • Stick to the Scope
  • Project Independence
  • New Development

Preparing the Team for Agile
  • Agile Roles
  • Co-Location is Key
  • Work Environment
  • Agile Coaches
  • Training Personnel
  • Tracking Your Work Artifacts

Preparing the Stakeholders for Agile
  • Who are the Stakeholders
  • Stakeholders Involvement Issues
  • Improving Working Relationships

Large Multi-Team Agile Projects
  • Scrum of Scrums
  • Special Teams
  • Multi-Team Processes

Developing Processes for the Enterprise
  • Release Phases
  • Employ Multiple Strategies
  • Requirements to Stories
  • Inclusive Modeling
  • Traceability in Agile
  • Communicate Priority and Progress
  • Adapt Software Build Processes
  • Mixing Agile and Waterfall Groups

Conclusion
Introduction/Summary

Although waterfall methodologies have not completely disappeared, they are struggling to remain a standard process for software development. As software becomes more mobile, complex, and innovative, and product versions require shorter release periods, the need to be more responsive to change has begun to work against the traditional waterfall processes which discourage changing requirements in the middle of a development cycle. As the waterfall process has become less and less effective for software development, agile has been gaining strength. In the current world of rapid software development, agile is the perfect fit.

When agile first began to take hold in the industry, there were many who thought perhaps this was just another fad. However, agile is clearly the hottest topic in the IT world today. Methodologies such as scrum, kanban, extreme programming, etc, have been around for some time. However, with agile’s recent rise in popularity, a greater number of large companies are either actively using or evaluating a move to adopting agile processes on their projects. Agile’s main principles include reducing documentation, lowering bureaucracy, using co-located teams, and better collaboration for cost-saving and quality assurance purposes. These principles have become more and more appealing, causing even larger companies to take notice. Industry studies confirm that a large percentage of companies are adopting agile processes and using agile techniques for their projects.

However, upon close inspection of these surveys, it is clear that those claiming to be agile projects clearly were not. Many companies will see agile is the silver bullet to solve all the usual woes of the heavily bureaucratic software development life cycle currently in place. Some may mistake the agile lean principles as reducing or abandoning processes which caused chaos and project failures. This can clash with the current IT community in a large enterprise which is used to the more traditional approaches like waterfall. Agile principles work well in small teams working in close physical proximity to one another, but they need significant changes to adapt and scale to the enterprise level processes currently found in place at large companies. Clear tailoring of agile methods needs to be done if they are to be successfully implemented in a large company.

This paper will examine some of the following processes to help large companies make the move to agile:

▶ Selecting your pilot project
▶ Bringing in the right people for success
▶ Acclimating customers and stakeholders to the new process
▶ Implementing multiple teams for large projects
▶ Developing hybrid processes for the enterprise

Start with the Pilot Project

As management of your large company, you want to get underway with your first agile project. So which project do you select?

To have a successful adoption of agile at your organization, it may be crucial to demonstrate success in one or more pilot projects. This means pilot selection will be critical in determining how agile adoption is perceived by the rest of the organization. With critics closely monitoring your progress, demonstrating success with a pilot becomes an important first step on the road to agile adoption. Companies with a large development shop and extensive methodologies already in place will be very resistant to change and will want proof that it can work. While a successful pilot project will demonstrate to the organization that the agile principles can work, a poorly selected pilot may not live up to the expectations from management which could hinder or even stop adoption of agile.

There are many factors to consider when selecting the ideal pilot project. Obviously, not every project is well suited to be your first choice. Mike Cohn, co-founder and current board member of the Scrum Alliance asserts four critical factors.
which affect the selection of a pilot project: duration, size, importance, and business sponsor engagement. In this section we will examine these factors and other guidelines you should take into account when trying to select your pilot project. It is very possible that you will not find the ideal project that fits all the criteria exactly. There will likely be some trade-offs involved, however it is better to start as soon as possible rather than waiting for the perfect pilot to fall into your lap.

**Pick the Right Team**

People are one of the most important factors for the success of any project. When putting together a team for your pilot project, be sure to select people who are on board with the new direction, as some team members may not be totally committed to trying out new methodologies. Make sure to select individuals who are willing to take the time to learn and apply these techniques the right way. Your pilot will not succeed if there are team members who do not believe in the direction you are heading. Those who are convinced that the current processes are fine and that agile is just a new fad will not be good choices for your first team. Negative voices on the team may start with one and spread to others as rapidly as wildfire.

If you are considering bringing in some outside consultants (scrum masters, leads, agile coaches, etc), make sure you hire individuals with the right kind of agile experience. When reviewing resumes, there will be those who claim to have agile experience but may not fully understand what it really is, or may simply have limited knowledge or experience with it. A coach who truly understands and has experienced agile methodologies would be ideal to assist in the process of forming your first team.

If you expect the pilot project you have chosen to be large, try not to grow to multiple teams too quickly. It is better in the beginning to start with just one team to focus on during the transition. As is detailed in this paper, coordinating multiple agile teams can be complex and not something you may want to deal with initially as it may increase the risk of failure. To increase your chances of success on the pilot project, you want to make sure you can invest the time in your first team to mature in the new processes before trying to deal with multi-team practices.

**Team Environment**

Try and locate or assemble a team in the corporation that is already co-located and avoid a distributed team. A co-located team is ideal for agile and will promote the concepts such as collaboration, face-to-face meetings, pair programming, and instant meetings as needed. Hopefully the culture of your company will allow for this. Traditionally arranged cubicles and offices will not work well for agile and e-mail communication is just not good enough. See if a war-room can be created or remove the cubicule walls in an area to promote collaboration. Let other teams in the company see what you are doing so they can ask questions and get excited about agile and want to get involved themselves.

**Size, Duration, and Scheduling**

When starting your pilot project with your brand new team, there is bound to be confusion. Team members will have just received training and will be unfamiliar with the new jobs they’re assigned to do. While learning these new processes, certain tasks they are used to performing will no longer exist, some will be completely different, and some tasks will be completely new. By starting small, confusion can be avoided and team members can spend time learning the new processes. If pushed with too much too quickly, team members may start reverting to the old ways of doing things and focus more on getting work done under the old structure rather than learning the new agile methodologies. Ideally a project should be small enough to be done by just one team so that your team members can focus on just learning agile.

Aside from team considerations, you also have critics in the organization to consider when selecting the project size and duration. If you select a project that is too small, then there will be skeptics who will say that agile only works on small projects. If the project is too long, then you may not be able to claim agile is a success until the end of the project and critics may see this as no different than a typical waterfall project that may run 9 to 12 months.
Therefor a middle ground is needed when selecting a project size for your company. Most likely that time frame may be somewhere between 3 to 5 months for your organization. This would be an ideal length of time because it gives your new scrum team time to adapt to the new agile techniques of writing stories, working in short iterations, demos, retrospectives, seeing the reward of this new methodology, and delivering software quickly. It is also long enough to show the rest of the organization that there can be success on shorter and longer projects.

**Organizational Support**
Successful adoption of agile, will require changes from the business side of the company as well. Having someone from the business side to work with your agile team as a champion, is critical to the success of your project. As agile may be a significant shift from the current business processes, an engaged sponsor can help the teams push back against these processes. As you deliver the expected results to your business sponsor, they will be invaluable to champion your cause and praise the new processes. Also, a sponsor spreading the word to others that a recent project used agile and delivered more than past traditional projects, will go a long way to convincing other business stakeholders to try this new approach for themselves.

Along with your business area, other departments at your company will have a say in the adoption of agile as well. Database and application support teams, help desk personnel, and others that you routinely work with for your application development will have a stake in the pilot as well. You will need their support to help make the pilot a success as processes change and procedures are adjusted to support your new agile techniques. Make sure to keep communications open with them because if the pilot succeeds and the company moves to agile, many organizational changes will need to take place. This is a great opportunity to start working towards integrating agile methodologies into company policies and procedures.

**Importance and Criticality**
When reviewing projects and their importance to your business, you may be considering selecting a project that is low risk and is generally considered of low importance from a business perspective. This way if your agile transformation fails, the stakes aren’t as high since the project was not critical in the first place. Resist the urge to select an unimportant project as your pilot. As mentioned in the previous section, you will need support from many different departments to make your pilot a success. The more important the project is to the business, the better support you will get from key personnel in different support areas. People won’t offer much support for projects deemed unimportant. It may even cause loss of critical expertise as management may decide to pull some of your team members and reassign to more “critical” projects.

Selecting relevant projects will help the success of agile adoption greatly. However, take care not to select highly mission-critical projects in your organization to try this new agile process. Agile has been described as a disruptive process innovation for a lot of companies trying to adopt this new paradigm. You need room to try new things, make mistakes, and learn from them. Unfortunately, this is something you simply cannot afford to do on a mission-critical project that depends on traditional processes already existing at the company.

**Minimal Risk**
Similar to adopting brand new technologies, care must be taken to select a project that is not too risky. If the project you selected is a complete throw away and has no business risk for failure at all, then critics at the company will simply say that it was a meaningless accomplishment. Although risk management is not explicitly addressed in agile, the methodologies manage risk implicitly and in the end will deal with risk more effectively than traditional methods. The pilot project should be risky enough to present an opportunity to highlight these risk management methods.
Avoid New Technologies
Since this is a new project, the team may be tempted to use some new technologies or architecture to implement your project. Since you are already changing software development methodologies, it is a good idea to hold off on using new technologies unless they are needed for the agile transformation itself. For example, if your team has never employed continuous integration or automated testing, they would want to invest time in learning these new technologies since they are part of agile processes. On the other hand, changing to a new application framework or a different language would be a bad idea and increase the risk of failure for your pilot project. Since you are trying to prove agile works for your company, avoid adding extra complexity to the work.

Stick to the Scope
It select your project, perhaps you have looked through a long list of business needs statements and high level requirements and finally chosen one that seems to fit well for your first agile project. Once you have, remain focused on the scope of the project you have selected from both the requirements point of view and for your agile transformation. If things start going well, you may be tempted to add other work in and jump into other projects or enhancements. Stick to the original scope of the project and the mission to deliver the completed scope after several months and show the success of agile transformation for your team.

Project Independence
Select a project that will have very few dependencies on other teams in the organizations. Since you are going to be the first agile project, other teams will still be following their current processes and methodologies. Trying to coordinate with these groups under your new agile processes will be difficult and add complexity to your agile transformation. For example, a project with interfaces to other applications will cause impediments and slow down your development processes.

New Development
If it is possible, try to choose a new software development project instead of an enhancement to an existing application. This will make it easier to incorporate a lot of agile methodologies into your iterations (unit testing, automated testing, continuous integration, etc). This will also let you experiment with shorter release windows that you may not be able to do with an existing application.

Introduction/Summary

Agile Roles
In a typical serialized development methodology (such as waterfall), software is built as artifacts are created and handed off between different teams. These artifacts go through requirements, design, development, testing, and finally deployment with each one of these stages creating a bottleneck. As documents move through each of these steps, some knowledge is lost and both money and time is wasted. It is often the case that by the time software is developed, there is a gap between what the customer really wanted and what the development teams understood and interpreted for the requirements.

With agile methodology, these bottlenecks are removed and hierarchies eliminated. All these areas in the life cycle are essentially present in the team because members are encouraged to become cross-functional. They are expected to help out in areas other than their own; if so needed by the team to accomplish the goals. This in turn helps reduce the need for formal documentation and sign offs as it promotes software over documentation as a core value of the agile manifesto.

Instead of specific titles, agile defines common roles for your agile teams. For each role, some considerations need to be made with regards to training to help make the adoption of agile a success. It is important to have completed training for all people involved before you start your first sprint.
Stakeholder - Your stakeholders are those that are potentially affected in any way by the development or deployment of your agile project. This is not necessarily constricted to just customers. There could be many other stakeholders involved, including users of the system, management, support staff, the portfolio, and also external systems that perhaps you need to interface with as part of development and/or deployments.

These stakeholders will work closely with the product owner on your agile teams to define the user stories that will form your product backlog and eventually be developed by your team. Typically these stakeholders would be involved in helping write business need statements and/or requirement documents to eventually be received by development teams. With agile, the development team will be constantly working with the product owner to define user stories and no longer creating detailed requirements. The team will be meeting more often, interacting with product owners and team members more often, and will be expected to react much more quickly to changes because of everyone’s increased participation in the project.

Although formal training is not typically needed, it would be prudent to have a high level introduction to the agile process and the changes coming to your company. This will help them to understand expectations and get them excited about the new process. This would be a good opportunity to inform them of the transition process and what will be expected. In the beginning, many formal meetings may be required but these will hopefully transition to more face-to-face meetings with product owners and agile teams as they progress through development iterations.

Product Owner - The product owner is the person on the agile team that is the interface to all the stakeholders and represents the customer. In many respects the job is very similar to that of a business analyst, but with the power to make more decisions without having to write all requirements up front. This person is responsible for gathering stories and acceptance criteria from stakeholders and maintaining (“grooming”) the backlog. This person will be the “expert” for team members when they have questions about stories being worked or need clarification from customers. This person also represents the team and will be continuously communicating to the stakeholders concerning status and demonstrations. Some other responsibilities include:

- Representing the needs of the customer and stakeholders
- Making changes to the backlog as needed that impact functionality or usability
- Writing user stories and scenarios
- Reviewing acceptance criteria with the team to help estimate
- Maintaining and prioritizing the product backlog
- Monitoring progress and making constant adjustments based on release planning
- Representing the teams efforts through communication and demos

Along with typical agile training your team will receive, some additional training may be needed. A product owner wears many hats including business analyst, business manager, and sometimes technical manager; thus additional instruction such as the Certified Scrum Product Owner (CSPO) training from the Scrum Alliance may be needed to handle the many facets of this role successfully.

Team Member - The team members are the individuals responsible for building the product according to the stories and acceptance criteria. Since team members are cross functional, the responsibilities include analysis, design, development, testing, planning and whatever other skills may be needed to deliver completed stories. Everyone may have different skills at their core (programmer, tester, etc), but as a whole team members will fill in the gaps as needed. All team members should have training in the core principles of agile processes. This is imperative before the first sprint begins since their day to day routine will be very different from what they are used. You want to start off with this knowledge and stick to it.
Team Lead (Scrum Master) - The team lead is sometimes referred to as the scrum master or team coach. This person is responsible for facilitating the team, removing any impediments, acquiring any needed resources, and making sure that they follow the agile processes. In some respects this is one of the most difficult roles to adapt to. Unlike a typical project leader, the goal of this role is to facilitate rather than direct. A successful scrum master has to work hard to empower the team to work on their own as a unit and as best as possible to get out of the way and let that process happen. In the long run an adept scrum master could disappear for days and the team would function without any problems having mastered the processes and operating independently.

Someone who is chosen to act as scrum master for a new team, should receive specialized training as well. Since the role is like a coach, it may be a difficult adjustment for someone who may have been a project/team lead in the past. They would now be asked to step back from directing the team and rather coach them instead on the agile processes to help them become independent.

Supporting Expertise - Aside from the roles above that are described in typical agile setup, there are certainly many other roles in your company that are responsible for the successfully delivery of you products outside of your agile team and business stakeholders. Some of these may include:

- Architecture experts to help the team if it is a brand new product
- Shared service teams that help with application builds
- Deployment personnel for moving to controlled environments in your enterprise
- DBAs to help with setting up and maintaining databases
- External test teams that may assist with integration testing

Although these teams may not necessarily need agile training, you certainly want to communicate with them on the new agile transition that your company and new teams are utilizing. Without knowledge and acceptance of the short iterations and changes in development methodologies, these support teams may quickly become impediments to delivering results rapidly. Many of the current processes (requests and approvals for deployments and external team support) will need to be changed to be able to support your new agile teams as more of your company heads in this direction. Keep these support teams in the loop and communicate often as some of the changes in these processes may be challenging.

Co-Location is Key
One of the values from the agile manifesto is collaboration. It is all about having your entire team working face-to-face to nurture the best work environment. Teams broken up by location and having to use emails, conference calls, and meetings to do their daily work will always struggle with efficiency. In most business offices today, cubicles and separate offices are still the norm. These stifle communication and prevent the full impact of collaboration from being utilized. Companies have to be willing to make many changes to the physical office layout if they want to take full advantage of all the benefits open collaboration has to offer.

Teams need to work together in large centralized areas. This allows such things as real time open communications, the ability to have meetings at an instant when needed without having to schedule conference rooms, and the common use of whiteboards and supplies. Being together in an open area also fosters pair programming as developers can simply move a chair and start working with someone at any moment.
Work Environment
Since co-location is the key to successful agile teams, you will have to adjust many things in the normal work environment to support collaboration. Obviously, dedicated space will be the first and foremost thing needed. The most effective agile teams have a dedicated area of their own to do their work. It is up to your management team to provide the resources the agile group needs. Below are some other common items in addition to the dedicated space that will help to make your new agile teams as efficient as possible.

Whiteboards - The more whiteboards you have for each of your agile teams, the better they will function. They are inexpensive and easy to obtain. Hang up large whiteboards on every available empty wall space in the team area. If each team member has a small cushion board, it’s a good idea to purchase a small white board for each of them as well. If needed, work with management to change any rules concerning hanging whiteboards. Make this a priority for your agile teams and put them in as many places as you can. You can even buy whiteboard paint or wallpaper to cover large wall areas which could be cheaper than buying the large whiteboards themselves.

Office Supplies - Most teams will require some of the simplest items as part of the agile process to get the job done. Make sure your teams have plenty of these items or have access to get them quickly. They should have quick access to dry erase markers, highlighters, index cards, post-it notes, pins for hanging paper, etc.

Large Table - Include a large table in a team area if needed. Perhaps your team has a large row of stations where their computers are located. A large table near them facilitates quick meetings, a place to brainstorm, or even share lunch together to foster more work discussion.

Wall Space - Make sure there is wall or cubicle space to hang important documents, schedules, help diagrams, phone numbers, etc. These should be able to be posted on walls as needed for quick access.

Servers - Some teams may want a dedicated server to do things like continuous integrations or setup their own application and database servers for speed and performance. If it will help the team to improve delivery, the company should order servers as needed.

Projector - Provide a projector to promote collaboration or discussion. When reviewing diagrams or perhaps tools like VersionOne or Rally to track your sprint at stand-up, a projector can help during these sessions to communicate. It can be hooked to anyone’s computer/laptop and projected on the nearest wall or whiteboard.

Camera - Many agile teams find the use of a digital camera invaluable. If your team just had a great working session and sketched out something on the whiteboard, they now need to take this information and type it in to a document or use for planning. Purchase a high-resolution digital camera that can be used by teams as needed for transferring images to their workstations.

Agile Coaches
One other role that will be helpful to your agile adoption will be that of the agile coach. This is someone who has experience in implementing agile projects and can help your company with that process. Some companies even bring in agile coaches early before the first pilot is selected to help with the strategizing the best way to roll out agile to the entire enterprise. They can help train your development teams, scrum masters, and even stakeholders in order to stay on track with agile adoption. The agile coach:

- Is an expert in agile methods and practices
- Can provide objective input without being influenced by company politics
- Is usually hired as a contractor outside the organization
- Has had experience implementing agile in several different companies of different complexities
The coach will understand how your agile teams and non-agile personnel across the enterprise will be affected as you move through the transformation. The coach can help management and teams move through this process and commit to the agile practices.

**Training Personnel**

Once your teams are selected and an agile coach has been contracted, it is time to plan the agile training process. Bringing in outside help to coach and provide training shows your management team that you are open to new ideas. Before any of your teams begin their first sprint, they should all have gone through training from a professional organization.

The coach can help you to decide what kind of training is needed for the different roles and how to incorporate high level overview with other teams in the company that will be affected by this transformation. Leave time for your coach to perform some assessments necessary to determine the knowledge base of the team since you may have some members with varying familiarity of agile principles already.

**Tracking Your Work Artifacts**

One last item to think about is the kind of tools that you will use to track artifacts for your agile teams and products. This will include user stories, acceptance criteria, tasks, team members, test scenarios, etc. Although there are many possibilities ranging from handwritten cards and notes all the way to off-the-shelf web based products to support agile, starting with low tech options will probably be the best solution before moving up from there if needed. This gets agile teams used to the process and as they progress and mature, they can make informed decisions about which tools will provide the detail and functionality to support what is needed in the organization.

**Low Tech (Cards And Whiteboards)** - Index cards are very important to the agile experience. They are the simplest tools used by the teams to track stories and progress throughout the sprints. Cards are easy and they let people work in parallel. They are extremely effective at communicating information quickly to people both in and outside of the team. The biggest drawback to cards is that you have to be physically located where the cards are for them to be of use. This means if your teams can’t be fully co-located, this solution will not work. Another drawback is reporting and retention of index card content. At the start of every sprint, a team member will basically grab a new stack of cards and start over. Unless they are entering this data into something like a spreadsheet, management has no way to track the progress or see historical data (trends, velocity, burndown charts, etc).

**Spreadsheets and PM Tools** - Using spreadsheets or project management tools is probably one of the easiest ways to track artifacts. Companies typically already have the software and you can find many free online templates for tracking agile team activities. It is fairly easy to find a template and customize it to suit your needs. Stories, tasks, and time can all be tracked in the spreadsheet or project tool. Although there is no logic or flow behind these, they can get the job done. If the entire team needs access, they will have to take turns updating the files from a central location which can become tedious but is manageable.

Being very flexible can also be a negative. If anyone can make a changes to the spreadsheet or project, it may break some logic. Additionally, if you have many teams using similar templates, chances are they will start to customize them for what works just for their team. Once that happens, consolidating this information into some high level reporting for management will become increasingly difficult.
**Dedicated Agile Tools** - Most tools now such as VersionOne and Rally provide support for most common activities needed for agile support. These include:

- Story, epic, and feature group management
- Release planning and sprint planning
- Tracking and reporting with story boards, task boards, and burndown charts
- Support for test scenarios with expected and actual results
- Defect entry and tracking in the backlog
- Integration with third-party tools for source control, requests, issue tracking, and automated testing

Especially if you are unable to utilize co-located teams, an integrated web based tool can provide support for both large and distributed teams. Even if your teams are co-located, you may have several teams working on one large project. A web based tool gives everyone the access they need to track the work on large projects of this kind.

You must also take into consideration the functionality and costs of these tools. An extensive review of tools would be needed before a company commits to evaluating whether a particular tool will meet the need and that costs are not too high. Most tools are licensed per user. Depending on the size of the company, this could be rather expensive so care must be taken to review and compare tools to find the best fit.

**Comparisons** - Low tech items that are more tangible and can be easily touched would be better served for initial agile implementations. Even for a large company that may eventually move towards web based tools, it is still a good idea for its agile teams to start with the low tech approach. It could be quite some time before the right tools are found and deployed in the enterprise, so it may work out well to utilize the low-tech options first. You should prefer the low-tech approach if:

- You are trying agile for the first time
- Your team is co-located

You should move towards a web-based integrated tool if:

- You have distributed teams in your organization.
- There are very large co-located teams
- Management needs real-time access to reports for planning and tracking

**Preparing the Stakeholders for Agile**

For a long time now, companies have instructed their stakeholders to follow a traditional (serialized) approach to software development. This starts with gathering requirements up front to be reviewed and approved. After that, the IT department will deliver a plan by which they will implement the requirements. After several months, the client is asked to be involved in customer acceptance testing and their product is delivered. From a customer point of view, there are only very short periods of communication with the IT teams in the beginning for requirement discussions and then at the end during testing.

The result is that there is a lack of confidence on the part of stakeholders that IT can deliver what they want. The 2011 IT Project Success Survey shows that only about 50% of projects implemented under traditional development frameworks are successful. Even with those that are successful, traditional methodologies show extremely poor effectiveness in areas such as quality, value, and ROI.

During the most important part of the project, the development phase, the customers barely communicate with the IT
teams unless there is a need for some clarification. Now, with new methodologies emerging like iterative, lean, and agile, we are asking our stakeholders to be continuously involved in the development cycle. This is a significant shift from what they have experienced in the past. As part of a new agile deployment, care must be taken to acclimate your stakeholders to their new roles.

**Who are the Stakeholders?**

Stakeholders can be defined as anyone outside of your team that has some kind of involvement in the outcome of your project. Stakeholders can include users, managers, customers, operations staff, support staff, staff working on other systems that interface with yours, and deployment support teams.

Clearly, there are many more stakeholders for a given project than simply the customers and/or users. To be successful with your first agile projects, all of these individuals must be taken into consideration when moving to new agile methodologies. Current processes in place will be greatly impacted and all those affected will have their own priorities and needs that must be taken into consideration. All these factors will affect how your new agile projects will succeed.

**Stakeholders Involvement Issues**

Under agile processes, stakeholders must now be continuously involved in the processes rather than in the brief intervals required by traditional methodologies. Stakeholder participation is very important to agile teams so that they can work to build projects that reflect the understanding of what is needed. From a scrum perspective, involvement during the actual iteration and not just the demos is required. Teams must work closely and regularly in a very collaborative process now to provide results that reflect the actual needs of the stakeholders. This reduces the risk that the functionality being delivered will be unacceptable and provides opportunities for immediate feedback.

As your agile teams begin to identify all the stakeholders and introduce them to the new processes, several factors may arise which will impact the level of participation from your stakeholders.

**Relationships** - Given that only about half of traditional methodologies deliver successful projects, the relationship between stakeholders and the IT department may already be strained in your company. This means stakeholders will be disposed to participate infrequently.

**Availability** - Many stakeholders accustomed to waterfall processes may already feel that it isn’t part of their job to be actively involved on a daily basis with IT. They are accustomed to handing off written requirements and not being involved again until testing time. This obviously increases the risk that the team will build the wrong thing since there are no frequent feedback cycles. In some cases there may be external factors that make stakeholders unavailable such as different working hours from the IT department.

**Communication** - Traditional methodologies emphasize formal communications and written documentation, over face-to-face and other informal types of communication. This can increase the overhead to the project teams which in turn increases the cost and time it takes to deliver functionality since changes go through the old gating process to affect change.

**Location** - Ideally when the team is close by or even co-located with the stakeholders, it becomes much easier for them to collaborate more frequently. If the stakeholders are in a different office, the team may only be able to meet face-to-face a few hours a week or less. If the team and stakeholders become more geographically separated, the chances for project success decreases.

**Level of Participation** - When stakeholders are not actively participating or are difficult to meet with, the team may be delayed by requests for information. If the situation does not improve, product owners on scrum teams will be tempted to anticipate the needs of the stakeholders based only on the infrequent meetings. These assumptions when creating stories and acceptance criteria could lead to potential defects or rewrites later on.
Improving Working Relationships

Given some of the issues and potential roadblocks when implementing changes to agile methodologies, it is important to come up with solutions to improve the relationships with the stakeholders and teams. The new agile processes affect the stakeholders and not just the IT department. The company will have to adopt a fundamentally different approach to planning, requirements, scheduling. These must now be done for each iteration rather than once in advance as with waterfall. With these changes, stakeholder involvement is still critical to the success of the agile project. There are, however, some strategies that can be employed to help improve the relationship and increase the chances of success for the agile project.

Get Stakeholder Support - Identify all the stakeholders who are going to help sponsor your agile adoption. Once you have identified the level of support from each stakeholder, you can determine who will help or hinder your agile adoption and see what ways you need to increase knowledge and identify possible risks to your support. Try to get support from the stakeholders themselves. Most likely, many of them will be open to trying the new agile methodologies once you educate them on the processes and demonstrate how quickly they will see results as compared to current methodologies. You also want to make sure there are processes in place so that current and potential future stakeholders can participate in the new agile project throughout its lifecycle.

Keep Stakeholders Involved - Once you have introduced stakeholders to the new agile processes, it may be difficult to keep them involved. In some cases, stakeholders will be pressured or diverted back to the normal daily jobs and lose focus on the new methodology. If agile adoption is done over a length of time at the company, some of these stakeholders may be overseeing waterfall projects along with the new agile projects. The tendency will be to fall back on the old ways of doing things (requirements documents, lengthy approval processes, etc). It will be vital for the agile teams to keep regular communication and involvement with the stakeholders so that they maximize the productivity of the teams. Without this continuous involvement, the agile adoption will be difficult and may eventually fail.

Adapt to Stakeholder Needs - Although it would be ideal to have complete access 8 hours a day to all your stakeholders, this is not likely. Agile teams may not always be co-located with stakeholders or there may be other factors out of everyone’s control that only allow a few stakeholder hours a week dedicated to that specific project. The agile teams will need to adapt to the needs of the stakeholders that are driving the project. This may mean setting up meetings according to their needs and schedules, extended trips to the stakeholders locations to accommodate them, etc. The more face time agile teams can coordinate with the stakeholders, the more successful the projects will be.

Include Operations Teams - Unlike a typical waterfall project which may have only a few releases a year, an agile project usually has short 2 week iterations and code is constantly and rapidly being transitioned through development, test, and production environments. This is a significant shift from the current processes and will mean the agile teams will have to rely heavily on operations staff, especially in the beginning to help change and adapt current company practices to accommodate these new practices. The support organization in the company needs to be involved in the agile transformation from the start so that they can learn the new systems being built, and work with the agile teams to adapt existing processes to the changes. Otherwise, the company's own support staff could become impediments to the agile adoption. It may even be beneficial to have operations staff spend some time initially as team members in agile to help develop these practices and take them back to the operations department to prepare for other agile projects.

Working with Senior Management - The senior management at the company plays a role in the adoption of agile. Since agile requires a continuous participation of stakeholders, you will need to educate senior management on the principles of agile so that they can support the level of involvement of stakeholders that is being asked for. Educate them on the benefits of agile, the technologies and techniques that will be used, and the implications. This way they can see the return on investment they will receive by requiring the stakeholders to be more involved and in changing the culture to move away from the traditional waterfall processes.
Large Multi-Team Agile Projects

In larger projects, companies may have teams much larger than the recommended 6-9 members per project. When the size of your agile team grows beyond this, it is recommended to divide up the teams into several smaller agile teams. One of the most effective ways to break up the development teams is around the different parts of the application being developed or supported.

With this method, each development team becomes responsible for one or more subsystems in the project enabling them to work as an effective agile team and to deliver all the features for their prospective subsystem. This also reduces the coordination required because the majority of communication would now be within the sub-teams. This allows companies to scale agile for these large projects since the teams are now basically groups of small sub-teams each applying agile methodologies to their subsystems.

Scrum of Scrums

Like the daily stand up meetings (scrum) in agile, a scrum of scrums is a communication tool used in larger projects containing multiple agile teams across a single project. The larger the project and the more teams that are working on it, the increased likelihood that there will be more impediments between teams due to overlapping work and inter-team dependencies. The scrum of scrum can be used to synchronize communication between teams and help manage and remove the impediments as quickly as possible.

Attendees - Each team involved in the project will select an individual to attend the scrum of scrums and represent the team. The team should choose someone who can best understand the issues of the team and discuss them. In most cases this would be the scrum master but it can be any qualified member of the team. The representative may also change over the course of the project. In the beginning, this may be someone who understands technical issues. Later when the project is in a formal testing phase, this may be someone who is an expert in testing issues.

When dealing with large projects that have several different kinds of supporting teams, it may be necessary to have representatives from some of the specialized teams as well such as architects, release managers, product owners, and project managers. If the teams are new to agile, then scrum coaches should attend as well to help keep the meeting on track. Like a regular scrum team though, care should be taken to make sure that there are not too many attendees at this meeting. Try for 9 or less so that the meeting is efficient.

Frequency - Unlike the daily scrum for an agile team, the frequency of the scrum of scrums should be determined by the teams and may not need to be daily. However, with a new project and new scrum teams, it would be best to have the meetings daily. Attendees would have stand up meetings within their own team and then attend the scrum of scrums to discuss any relevant issues for that day. Although a scrum of scrums should be limited to fifteen minutes, more time may be allotted. Issues brought to this meeting should be resolved right away if possible, so more time may be needed for discussion and problem solving. Over time, the scrum of scrum meetings can be longer in duration and held less frequently, such as 2-3 times per week.

Agenda - A typical agenda for scrum of scrums is very similar to the standard agenda for an agile daily scrum. Since multiple inter-dependent teams are involved, an additional 4th question is now introduced:

- What did our team do since we last met?
- What is the team working on next?
- Are there any impediments to the team that need to be addressed?
- Are you about to put something in another team’s way?
This last question is extremely useful for multiple teams. By notifying other teams here of potential upcoming issues (major code check-ins, architecture changes, environment outages, etc), teams can plan and schedule around the issues and minimize the impacts.

On an agile team, a daily stand up is to inform team mates of what is happening and to discuss impediments; it is not used for problem solving. However, the scrum of scrums is different in this regard. If a problem or impediment is brought up for discussion, then it should be addressed immediately in the meeting. Since issues at this level could impact several teams and many individuals, it should be resolved right away if possible.

**Special Teams**

For large development projects requiring multiple scrum teams, other special teams need to be considered. With multiple teams having to work together on a related project, special processes and coordination need to be put in place to effectively communicate between the teams and deliver the work efficiently. There are some teams other than the typical scrum development teams recommended for large agile projects to help coordinate these efforts. These teams would form their own meetings or “Scrum of Scrums” to discuss related issues.

**Project Management Team** - This team is similar to a program management team. It is made up of leads and coaches from the various other scrum teams. The main focus of the team would be to deal with the management aspects of a large project above the development level which can include resources, clients, dependencies between teams, and outside factors to the project. This team would meet briefly on a daily basis to identify and resolve any issues related to project management.

**Product Owner Team** - A product owner team would be made up of the product owners from each of the scrum teams. In a large project with many stakeholders divided up into a variety of teams, this team of product owners would work with the body of stakeholders to gather requirements (“stories”) and decide how best to distribute them amongst the teams. If issues arise as to which teams work on which requirements, the product owner team can resolve this by assigning stories or in some cases breaking them between teams for parallel development.

**Architecture Team** - For a large project in its beginning development stages, architecture plays a key role for development teams. In this case, an architecture team should be formed to identify the strategies for the technical implementation of the project. This team can be comprised of architects and senior staff from each scrum that would meet to decide on these issues and guidelines for implementation. The team would focus on items such as application development standards, interfaces, database, and other technical implementation. The team would meet on a regular basis initially several times a week, which may reduce over time as the system matures.

**Shared Services Team** - A shared service team (or “Release Management” team) is responsible for the activities surrounding management of deployments of software through different environments. This team would coordinate the deployment and builds of the software from all the teams from development, test, and finally production environments. This team would consist of build engineer with representatives from other areas such as development, testing, and operations. The focus would be on maintaining accurate build and deployments, and working with operational teams to deploy the entire product from all the packages of sub-teams to one product.

**Testing Team** - For larger projects, an independent test team will be needed to do more formal and complex end-to-end testing between the components of the application, in addition to testing any external interfaces to there systems. This team may involve testers from the scrum development teams along with other testers outside of the group. This team may be engaged to do more formal system and customer acceptance testing comprised of several iterations of work for a large release. They would validate functionality above and beyond what is done by the scrum teams during the iterations along with interfaces to other systems which may follow similar release schedules. This usually works well in companies adopting agile that still may need to interface with projects and waterfall schedule for coordination.
Multi-Team Processes
The scrum of scrums for multiple development teams, affords the opportunities for representatives from each team to talk about issues surrounding the day to day development during iterations. However, with multiple teams, some other processes besides the daily stand-up can also be implemented to improve processes. This is in keeping with the agile motto to “inspect and adapt” and can be scaled for multiple teams as well utilizing some of the same meetings/processes of a single scrum team.

Planning - Planning is constantly happening in agile, sometimes called “grooming the backlog”. If there are multiple teams on a project, there should still only be one backlog for the application. Like a small team, the product owner team on a large project mentioned earlier, would constantly be meeting with each other and the stakeholders to “groom” the backlog and plan the sprints. Planning for releases would be followed as well with all product owners working together from different teams.

Retrospectives - In scrum, the sprint retrospective allow the team look for ways to improve the processes. This same process could be applied to multi-team retrospective after each sprint. After each team conducts their retrospective meetings, they can then all meet together for a common retrospective. Teams can discuss issues that affect everyone a high level and then choose action items to implement in upcoming sprints.

Combined Demo - Constant demos to stakeholders is a big part of agile. However, given a project with multiple teams, it makes sense to combine these into one large demo. Although a bit more formalized, this allows all stakeholders to interact with all teams and see different parts of the system as one demo. The product owners and at least 1 or 2 members from each team should attend these formal demos. As each team demos the stories from the last iteration, feedback from customers is heard from all teams. This allows items that may affect multiple teams to be discussed and captured as possible future stories at one meeting. Another advantage of a large combined demo is availability of stakeholders. Rather than multiple teams trying to find open time on a calendar, one single meeting can be setup so all stakeholders can attend and see different parts of the system demonstrated from all the teams.

Developing Processes for the Enterprise
As you begin to integrate agile methodologies into your enterprise, you will discover that agile and waterfall processes don't always work well together. Mature organizations will have many tight controls on the development process, including tollgates or other management control points that dictate how and when software gets released to production. Most organizations shy away from frequent production releases because of these tight controls or simply because the lengthy and continuous waterfall processes simply can not support rapid releases.

Agile methodologies, however, differ in regards to releases by completing frequent sprints (“iterations”) with releasable code held as one of the main goals. Agile focuses on releasing software as quickly and efficiently as possible without sacrificing quality. At first glance, it may seem that existing waterfall processes would be in conflict with the agile methodology. However, it is possible to have agile processes that live within a waterfall release approach.

It is the creation of this mixed or hybrid set of processes that a company adopting agile will have to define. While initially it may seem that these two worlds will not work well together, it is definitely possible to come up with a model that will be successful for your organization.

Release Phases
Although agile processes utilize many of the standard lifecycle phases during sprint iterations - requirements, design, coding, testing - most applications don’t really deploy software to production after every iteration. Due to external factors in the organization such as restricted release windows, most companies will have only a few software releases a year even while using agile.
At a high level, projects have three overall phases: planning, development, and test and deployment. A typical agile approach is to bookend these phases around the main development iterations used by agile. Development teams can certainly take advantage of agile approaches for improved quality, quicker development, and high customer satisfaction, while over time, incorporating the other project phases into their iteration framework.

**Planning (Sprint 0)** - Before the agile teams begin to code, it is best to take time to identify the goals for the project and to prepare any needed support for your teams. Traditional waterfall processes will usually spend a large amount of time with initial planning, where agile requires that very little effort be invested in big, up-front project planning and formal requirement writing.

For any projects that are getting started with new teams, there is a need to have some time up front to get the project moving in the right direction. In agile, this is sometimes referred to as “sprint 0 (zero)”. This initial sprint for planning may involve such activities as:

- Developing initial backlog with stakeholders
- Planning high level architecture
- Configuring development environments
- Defining build processes
- Defining the initial release plan

This initial sprint may be somewhere between a few days or weeks depending on the size and complexity of the initial project and the number of different groups involved. To avoid delays, sprint 0 needs to have stories that are deliverable. Once the team has completed all tasks and received the needed support for the project, they can move on to the initial development sprint.

**Development** - During the development phase of the project, the agile teams should be building the functionality as described by the backlog which should already have been created during the initial sprint. During development, the agile teams would be going through the iterations (sprints), building out the functionality required, testing code, and demonstrating that functionality to the stakeholders for their acceptance. These iterations are usually small (between one to three weeks) with most organizations using two week sprints as the most common.

The goal for each iteration is to deliver working software to the customer for release at a pre-determined time. Within each iteration, the typical lifecycle phases of any software development are done including:

- Design (as needed for large stories)
- Development, refactoring, and unit testing
- Regression testing and continuous integration (CI)
- Stakeholder demo of stories meeting user acceptance criteria and team definition of “done”
- Continuously updating the backlog and release plan

These iterations would continue until enough of the software has been built (based on the initial release plan) to justify packaging up the software and moving to a formal release test stage.

**Test and Deployment** - Although agile teams will have unit and regression tested the functionality during each iteration, most organizations will still need a formal QA or CAT test phase to be implemented prior to the release of code to production. This is especially true when your agile application must integrate with other waterfall applications that are still following a planned release schedule for design, development, testing, and deployment.

During this formal test phase, the software developed from all the sprints for a particular release would be bundled up and delivered to the test environments for formal application and interface testing that is standard within interdependent systems. Once the formal test phase has been completed, the software can be deployed to production.
While this formal testing phase is in progress, your agile teams should already be at work in the planning and development phases of the next scheduled release.

**Employ Multiple Strategies**

Agile itself is not a methodology, but a framework with several guiding principles. The agile manifesto describes a simple list of guiding values to software development:

- Individuals and processes over tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Along with SCRUM, there are many other methodologies and practices based on iterative development that can be used as part of an agile development framework. Your organization should examine these strategies and methodologies to see which ones best fit each project and functionality being delivered. Since agile is not a methodology, it is up to the IT organization to examine the available agile practices and create a methodology from these that works for their organization. Here is a brief description of some of these methodologies and practices.

**Scrum** - Scrum is essentially a project management methodology for a simple and repeatable way of managing work which can be used for both new projects and ongoing maintenance. It contains a small set of practices and predefined roles. Scrum is becoming the de facto standard for managing agile software development projects. It is guided by the principles and values of the agile manifesto which emphasizes people over processes, working software over documentation, collaboration over contract negotiation, and responding to change over following a plan. Since scrum consists of only a few practices it usually does not contain sufficient guidance, so other methods such as XP would be applied for development teams.

**Extreme Programming (XP)** - Extreme programming (XP) is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation. Some of the highlights include small releases, pair programming, test-driven development, and continuous integration. It is often seen as complementary to Scrum.

**Feature Driven Development** - Feature Driven Development (FDD) is an iterative software development methodology intended for use by large project teams. In FDD features are defined using user stories similar to Scrum. They are usually expressed in the form of action, result, and object. Some examples might be “Calculate the sales tax of a purchase”, “Approve the submission of a load”, and “Authorize access to a function”. Like stories in scrum, they are the source of requirements for the project.

**Test-driven Development** - Test-driven development (TDD) is a process of using automated unit tests to drive the design of software. This technique requires the developer to write the automated test code first before the actual application code. In fact, the purpose of writing application code should be to satisfy unit tests that have already been developed. This ensures that all developer code is tested. It enables retestability of code at any time since the changes are a part of the code and can be automated. At any time, builds can initiate all the automated unit tests in the software to make sure that all components are still working properly.

**Continuous Integration** - Continuous Integration (CI) is the practice where developers on a team frequently integrate their work by checking in code to a main repository several times a day. This approach will reduce integration issues with the software and catch issues quickly. It can also be used in combination with automated unit testing from the practice of test-driven development. As changes are committed to the repository, CI prescribes the process of automatically triggering build and the execution of the automated tests to make sure any changes do not break software before send to the main line of code.
**Kanban** - Kanban takes the development process at the organization and visualizes the status of the current work. It provides a method to constantly adapt processes to work out any issues for new work as it arrives. It uses a work-in-progress limited pull system to expose any issues in the current process and help improve it.

**Agile Unified Process** - For those organizations who invest heavily in IBM’s Rational Unified Process (RUP) as part of their software development processes, the Agile Unified Process (AUP) may help the overall adoption of agile. AUP is a simplified version of RUP using agile techniques and concepts. AUP applies agile techniques including test driven development (TDD), Agile Modeling, agile change management, and database refactoring to improve productivity. Compared to RUP, there are only a small set of disciplines used in the methodology to make it lean.

**Requirements to Stories**
Sometime after you have moved from waterfall to agile, you will discover that many things have improved resulting in quicker development cycles, team strength, and better practices like continuous integration. One of the concerns companies have at this point is the management of requirements. As your scrum teams work closely together with their respective stakeholders, some other areas in the organization may feel like they are not having enough involvement. Within the organization the process may appear a bit erratic because of the speed at which teams are now working to develop stories. The product management side of your organization also needs to acclimate to agile which may be even more difficult than the rest of the team.

Now that your IT teams are agile, they have moved away from generating large requirements documents to writing epics: overall conceptual needs that then breakdown into small, deliverable user stories. They are having more face-to-face meetings and whiteboard sessions with stakeholders to refine the acceptance criteria for these stories. The downside to this is that the lengthy approval process that you used to have for many other stakeholders to review, provide input, and approve the written requirements is now gone. Any valid concerns about the requirements are now being missed because these stakeholders are not involved directly with your team nor do they have exposure to your backlog of stories.

Although the requirements reviews the organization used to conduct are not part of an agile process, you may want to keep this process in place with slight modifications. Make it a practice to still get together with all the stakeholders and talk about the features you are going to be implementing. Review the epics and stories in place of the lengthy requirements documents that used to be reviewed. Also, conduct architecture reviews of the new features you are developing with other teams in order to get the needed feedback for your development. In most cases, these teams are not going to come to you and read the backlog of epics; you will have to seek them out and bring them together to be successful in your agile project.

**Inclusive Modeling**
One of the core values of the agile manifesto is “individuals and interactions over processes and tools”. This is the basis for a technique in agile practices known as inclusive modeling. This technique is used to explore requirements with your stakeholders and customers in a simple, interactive way.

To obtain the detailed acceptance criteria for stories, the scrum team would work with the stakeholders to explain what they mean. This can be done in many ways including drawing on paper or on large whiteboards. This becomes a much quicker and accurate way of discovering the details than old methodologies in which developers would go off and write complex and lengthy design documents which may not be exactly what the customer wanted. After all, the customer is the expert in that domain. If you truly want to follow the core value of individuals and interactions, your agile teams should adopt these processes of modeling and design.
Traceability in Agile

Most companies today have IT processes which include creating a requirements traceability matrix (RTM). A formal traceability matrix often evokes strong response from the agile community. When talking about lean principles in the agile world, the RTM is considered to be excessive and most likely not really needed. For companies that are used to having this document created as part of any software development requirements, it may be difficult to eliminate this artifact completely from your current processes.

Creating and maintaining this artifact can be extremely time consuming; particularly when it comes to demonstrating traceability. Since traceability is simply the means of demonstrating that your software met all the requirements asked to be build, one should consider other means besides the RTM.

As part of the move to agile and other lean practices, people should ask for the reasons behind the maintaining of the RTM. Ask why management wants such traceability, how often they will need it, and most importantly what information they will actually accept. Since agile artifacts such as stories containing acceptance criteria already have the traceability built in, management could be convinced to accept other automated formats; especially if you explain the cost savings of not having to manually pull together and maintain just for the sake of process. Your agile process will contain artifacts to represent traceability (stories, historical data, source code, design docs, test scripts etc), and these should be used when possible for traceability.

After analysis is done with your new agile methodology, your company will probably discover that the total cost of maintaining these matrices will outweigh the benefits compared to using the existing agile artifacts. Use this information as a means to convince your stakeholders and management of the cost savings and let them make the final decision.

Communicate Priority and Progress

Under the waterfall methodology, how work was prioritized and implemented was important to delivering working software in a timely manner. This is equally important with agile. However, now that your agile teams are communicating more frequently with the correct stakeholders, the team will need insight into why work is being prioritized by your product owner the way it is.

This is where the sprint review or demo now becomes an important part of the process with your stakeholders. This is not only the chance to review the work you have done in the last sprint, but also the chance to review the upcoming sprints and which stories are being implemented, in what order, and why. The upcoming sprints can be reviewed with the stakeholder and approved.

Reviewing the completed stories from prior sprints along with the upcoming work, also presents an opportunity to communicate the overall progress of the entire project. The agile processes should make the progress of the project much more obvious to your stakeholders since they can clearly see what functionality has been completed, what functionality remains to be done, and any outstanding issues.

Adapt Software Build Processes

In a standard waterfall methodology, there are points in the release cycle where software builds are executed. Typically, this starts in development where a team lead or perhaps a specialized team may execute an integrated build perhaps every few days or weekly. In some cases, there may not be a first integrated build until preparation for the first system tests begin. During this period, developers will typically create builds on their local machines only and may not integrate or check in code for some time or until just before testing starts. This means developers and the build teams will be running into integration problems late in the development cycle, scrambling to correct issues before the first round of testing can begin. In a waterfall cycle there may have to be time afforded for this correction because of the interval between builds, but not with agile.
Under the agile approach, software builds are executed much more frequently. In the beginning, this may be at least daily and as teams adopt continuous integration into their processes, builds will happen several times a day each time code is checked in. This continuous integration process allows teams to discover immediately if the code they have modified has adversely affected some part of the system. Code merges and automated tests such as JUnit would execute at build time and report back issues. Developers must correct the issues immediately since the build will be broken until the item is corrected.

This new approach in agile will mean significant changes in the organization on how build processes work and how teams are structured. Companies will need to invest in acquiring hardware and software to implement continuous integration servers and automated build processes which are triggered by check-ins. Teams which currently handle builds will need to be trained on setting up and configuring this software with new build processes to support the agile projects and teams.

**Mixing Agile and Waterfall Groups**

As your company adopts agile across many projects, there will be challenges as it clashes with existing waterfall groups. Depending on the size of the company, it may be years before you can fully integrate agile into the entire IT enterprise.

The waterfall methodology promotes big up front (BUF) requirements gathering and design which leaves little room for change during the development and testing cycles. Contrast that with agile, which promotes smaller iterations, less time up front, and the ability quickly change requirements and directions at the next iterations. Not only will there be inevitable clashes between development teams supporting agile and those still supporting waterfall projects, but there will also be support teams who may have to deal with the different processes simultaneously. Strategies need to be in place to help bring these methodologies together during the transition phase and introduce the agile methodologies to other teams still utilizing waterfall to help promote the advantages and make the transition easier.

**Use Agile in the Development Cycle to Help Transition** - To help with the transition to agile, one strategy is to employ the scrum methodology during the development and application test cycles. This encourages architecture and design to be defined at the beginning with formal integration testing and deployment at the end through existing processes. This allows teams to still take advantage of lean methodologies during the development cycles and still show progress to stakeholders after development iterations. As groups outside the scrum teams begin to see the benefits, more and more lean processes can be adapted for these beginning and ending phases as more and more teams transition to agile.

**Manage Scope** - With waterfall projects, scope is managed through change control processes which prevent scope creep from affecting release dates. Under agile and with quick iterations, stakeholders may get the wrong impression that anything is possible and they can change at a moments notice. However, scrum masters have to realize that the iterations are marching towards an agreed upon goal which is part of the release plan for your agile teams. Stakeholders swapping new work for work already completed in prior sprints is be definition scope creep. New work is entered into the backlog and then prioritized for the next iteration.

**Don't Release Too Often** - Under agile, working and tested software is ideally accepted by the customer at the end of each sprint. This doesn’t mean that you are releasing to production every sprint. In many cases, for large applications, it may make more sense to schedule releases after several sprints. This allows time for formal integration testing, especially when the agile project may be dealing with a system still on waterfall schedules and the releases must coincide.
Conclusion

Agile is a mature methodology that clearly can deliver better results compared to the waterfall approach. Many organizations are already using agile principles or are planning to move to agile in the near future. Companies will find that the concepts and methodologies from the agile manifesto (small iterations, co-location for face-to-face meetings, self-organizing teams, etc) have to be adapted to work for their current culture and processes. During the transition to agile, you will also have to adapt many processes to mix waterfall and agile processes in order for them to successfully coexist. There will definitely be challenges to scaling agile for your large development projects. Commitment to scrum and lean principles in combination with help from experienced agile coaches will play a huge role in the success of agile adoption and tailoring processes to fit your company.