Effectively Using Agile Methodology to Deliver Packaged Software Solutions
# Table of contents

- Introduction ................................................................................................................................. 1
- Common Misconceptions About Agile ......................................................................................... 4
- Key Factors In Implementing Agile Successfully .......................................................................... 5
- Looking Ahead ............................................................................................................................. 12
Introduction

Agile is a widely accepted software development methodology. It is evident that the market continues to demand increased quality and speed-to-market while containing implementation costs. Flawed implementations of Agile can result in a failure to completely realize its full benefits. This can lead to a misperception that the methodology itself is not adequate, when the true failure is the implementation, not the design of the methodology. Organizations that are new to Agile will reap significant benefits from involving resources who have experience that has been cultivated through a first-hand understanding of drivers of success and sources of pitfalls.

1. 2011 Standish Group CHAOS Manifesto
2. Results from Scott Ambler’s 2011 IT Project Success Survey posted at www.agilemodeling.com/surveys/
3. QSM Associates ‘Agile Impact Report’
Replacement of a core system can be a daunting task to an organization. Implementing a packaged solution allows the organization the opportunity to implement a customized, full-scale solution without embarking on the endeavor of a full ground-up build. An organization that has chosen a packaged solution must carefully consider the development methodology to be employed. Agile is particularly well suited to a core transformation project that is centered on the implementation of an “out-of-the-box solution.” Having baseline functionality supports the need to allow for meaningful demos even in early sprints, even if the development effort only shows marginal changes from the base product.

The key benefits of selecting a packaged solution may include:

- Speed of implementation with base functionality already in place
- Stability of an existing platform
- Customization through configuration rather than having to build ground-up
- Quicker engagement of subject matter experts
- Earlier engagement across the organization

These benefits that were likely significant drivers in selecting a packaged software solution can be easily eroded by the pitfalls of outdated development methodologies. It is difficult for the subject matter experts to have a full appreciation for the full end-to-end functionality of the “out-of-the-box” product from the beginning of the project. Traditional methodologies would essentially require this end-to-end understanding before work can begin. Agile is uniquely suited to allow the organization the opportunity to gain understanding of the base product while they work toward configuring the full solution. It is critical that the organization is carefully guided through the Agile process to avoid many common pitfalls that can introduce delays and missed opportunities into the project.

Implementing the Agile process itself is definitely iterative. Just as the members of the team may not have a full understanding of the packaged solution, those who are new to Agile will not have a full appreciation of how it works from the beginning. The team will learn and improve upon the process through each sprint. Experience is the key to avoiding dangerous pitfalls in implementing the Agile process. Make sure you surround your team with plenty of resources who will bring their experience to the table to keep the team on track. Seek out implementation partners who can demonstrate a proven track record of success in not only “doing” Agile, but also “being” Agile.

For organizations or teams new to Agile, the learning curve can be steep. The more mature the organization, the greater the challenge may be. The diagram on the next page illustrates a typical progression of this learning curve.

You will no doubt see clear behavioral changes as the team adopts the new ways. Wherever the team fits along this development, make sure that movement does not stall. Be diligent in your efforts to move as quickly along this continuum as possible. Adding members to the team who have successfully been down this path before will no doubt accelerate this process.
Behavioral Characteristics as Agile is Adopted

Newbie
- Individual contributions
- Work is finished, then “over the wall”
- Re-visiting “old work” is painful
- Only accountable for my work

Learning
- Sharing
- Some upstream/downstream involvement
- Ok to re-visit some items
- If I have time, I can help

Fully Adopted
- Fully collaborative
- No restrictions
- Iterations become the norm
- We all own the outcome
Common Misconceptions About Agile

Many misconceptions exist around the use of Agile. These misconceptions can undermine a quick and successful adoption within the organization. They can be quickly debunked if you have the guidance to implement Agile properly, while holding true to its principles. These misconceptions should also be clearly addressed as part of the team’s Agile training.

- **Agile means you can change your mind as often as you like.** As the team iterates through multiple sprints, needed re-work from previous sprints will certainly be discovered. These discoveries should not be viewed as distractions, but rather embraced as an expected and necessary component of continuously improving development work. Identifying the line between improvement and scope creep, however, can be challenging. While iterating through improvement cycles is expected and should be accounted for within the project plan, the work from previous sprints cannot bring the current sprint’s work to a grinding halt. The project team should establish scope boundaries, and leverage quarterly targets to avoid scope creep.

- **No documentation required.** This is definitely not the case. Even though a complete set of requirements will not be delivered prior to development as they would be with other methods, documentation is still critical. All members of the team must be able to understand changes, when they occurred, the status of development, and the outcome of testing. Documentation need not be exhaustive, but tracking the progress of development and changes is critical. Full traceability is still required and can be leveraged through the use of tools such as internal intranet pages or Sharepoint. The key is effective versioning and workflow so that all can easily understand the status of each requirement and subsequent change.

- **Agile will guarantee success of the project.** Agile is not a silver bullet. Resource skills and availability are still key. Make sure that you have the right people in the right seats. The organization must be prepared to muster the right amount of resources as well. If the organization does not have the right number or the right skills, outsourcing may be a great option to quickly bolster the team. The organization's level of maturity can certainly play into this evaluation as well.

In addition to numbers and skillsets, the team must be able to have decisions made in an efficient manner. Lack of decision will rapidly turn into lack of direction. Quick turnaround on development work can quickly be hindered if decisions about requirements can’t be made quickly. Be certain that your SME’s either have decision making authority or have ready access to those who do have the authority.

- **We don’t test in Agile.** As the team iterates through development, testing is just as critical in Agile to assuring the quality of the delivered product as in traditional methodologies. Testing within the Agile cycle allows for defects to be identified in much closer proximity to the development work. This in turn leads to more effective defect resolution. The key here is to keep test cases up to date as requirements evolve. Once again, the tool that is used for versioning and traceability plays a critical role.

- **Project planning isn’t very important.** Planning is critical from the outset of the project. Design can and will change as the project evolves, but a high-level design is critical from the start. Development work on the project should not begin until a solid sprint plan is created. Without this, you will struggle with moving forward in a methodical fashion. If the sprint plan is weak, you may suffer from challenges with resource allocation. Make sure that you have solid buy-in to the high-level design of the product that you will deliver.
Key Factors In Implementing Agile Successfully

The driving factors of implementing any technology project will be defining the requirements and then determining how best to leverage the technology to support those requirements. The diagram below illustrates the intersection of the agreement on the requirements with the agreement of the technology solution.

Source: Complexity and Creativity in Organizations, Ralph Stacey

Agile will, by its own design, have you starting at least some distance from either axis in most cases. Careful attention to several key focus areas will assist in the drive toward agreement on requirements and certainty in technology. This will also mitigate the risk of the project missing its targets.
Governance and Stakeholder Management – While the risks of the development team not fully adapting to the new methodology should be readily apparent, don’t overlook the importance of all of the buy-in of all of the leadership. The project is dependent on a deliberately defined and carefully followed governance model. Everyone must participate in driving Agile through the organization. This really must be a cultural shift for the organization. In order for that shift to take firm hold, the members of the leadership team must become fervent champions of Agile.

“By focusing on the methods rather than the culture of Agile, we set the expectations of everyone, from the executive suite to the PMO to the business stakeholders, that all we have to do to become Agile is migrate from Work Breakdown Structures (WBSs) to backlogs and from Gantt charts to burndown charts, and we’re on our way to The Agile Enterprise.” Rick Freedman, "Agile Grows Up and New Challenges Emerge."

Deliberate Planning – Laying out the sprints and the associated stories within each sprint is the foundation for success. Failure to do this will almost certainly result in scope creep. Failure to plan is planning to fail—who said that? If you haven’t clearly defined the boundaries so that everyone knows what is in and what is out, your estimates and budgets will not hold. Sequencing the sprints in logical building block order will also improve efficiency. Make sure the high-level decisions and components are in place before you jump into the details. After all, your early demos will almost certainly show high-level components.

Sprint planning meetings at the beginning of each sprint are the perfect forum for re-evaluating upcoming work. Re-prioritizing work should be expected. As part of a collaborative and cross-functional process, some tasks may need to be pulled up from future sprints while others are delayed until later sprints. This shuffling of work should not be viewed as a failure of the plan, but rather a successful use of the Agile methodology.

Talent Assessment – Each developer brings to the project a unique set of skills and understanding based on previous experience. It is imperative that the team be trained in common methods of development. The themes that need to be covered here may vary a bit based on the individual project, but will likely include topics such as coding syntax, documentation, and naming conventions for data model extensions. Consider pairing an
inexperienced developer with a more experienced one early in the project to reinforce the approach and methodology. Your project plan should include an allowance for adequate training at the onset of the project.

Dependencies between and among the workstreams must be considered. The individual project teams will not be working in silos. While a detailed project plan must be created, the plan itself must be agile as well. Individual components may take less time or more time than planned. Resource availability may also not match the original plan. You should be prepared to shuffle a bit as the project proceeds. The plan can and should be iterative as well.

- **Ongoing Training** – Every project brings together resources with a wide variety of experiences. Some may have technical strengths, others may have great project leadership abilities, and some may have great business acumen for the subject at hand. Agile will allow each resource the opportunity to gain knowledge by learning in small bites. Each building block of the project adds to the understanding.

Limitations of inexperienced resources can cause a project to lag in the early stages as everyone gets up to speed. The lack of experience can lead to re-work, but the iterative process that is the cornerstone of the philosophy supports overcoming this challenge. The key here is proper planning. If you have the ability to assess your resources’ skillsets early in the project, plan a bit of redundancy around initiatives covered in early sprints based on perceived weaknesses. Make sure to allow for proper training in the use of the Agile process from the beginning of the project. Even with resources experienced in Agile, take the time to make sure everyone has a common understanding of how each project will be managed. Sufficient time should also be allowed for training of resources that join the project after the initial kick-off training.

Communication both within and across teams and workstreams will help to get resources up to speed more quickly. Use the talents and skills of the team members wisely. If you find that a resource may be more valuable on one team over another, don’t be hesitant to shuffle. After all, the philosophy calls for development to be an iterative process—your team structure should also employ an iterative approach where needed.

- **Solid Architectural Foundation** – Just as you would start the process of building a house, your first sprint is one that requires you to establish a foundation also known as sprint 0. During Sprint 0, or Inception, it is imperative to develop and agree on an overall architectural vision for the project. Make sure that you include the right resources so that your plan fits into the overall architectural vision for the organization. For this planning portion, you may need to involve people outside of the project plan to gain input and assist with buy-in. Improve overall productivity and reduce development time by thinking through some of the critical technical issues up front to avoid fruitless technical paths and to minimize refactoring. All of this should be done with an eye toward building a system that can be efficiently maintained, modified, and upgraded.

Planning for an efficient process for builds and deployments will streamline your process. The backbone of Agile is regular delivery of work product through your development and testing environments. Don’t let the deployment process drag down your ability to deliver work. Deployment schedules should be widely published so that everyone knows the timetable.

Each workstream will continuously modify the data model. A regularly scheduled review of these changes across workstreams will help to avoid re-work. It is also a great opportunity for members of each workstream to gain a better understanding of work that is being done across the team. Depending on the size of the project, these updates should occur on a weekly or bi-weekly basis.
Team and Communication Transparency – It’s not at all uncommon for the end goal to get lost quickly in the daily and weekly task-level focus of the project. Look for opportunities to keep the team engaged in how the daily work moves the project ever closer to the delivered solution. Transparency about the shortcomings of the current system, along with a continued communication about the direction of the project, will not only help to reinforce the objective, but will solidify that the work being delivered will actually be the cornerstone of the solution.

How often do you see a large group of project team members – both internal and external resources – huddled in a conference room or in some secluded corner of the office for months on end while members of the organization frequently ask each other “What is that army of people actually doing?” Properly implemented, Agile not only allows you to actually show something tangible after just one month of development—it requires it. Breaking the project into smaller, easily digestible bites allows this happen.

Make sure that the first pieces of functionality actually are tangible and not just some basic foundation work. This is your opportunity to get the organization excited early. Be prepared to show something more than just a basic administrative function that most users won’t use or see. At least some of the initial work should tackle some of the more complicated issues. Try to follow the users’ workflow. Perhaps show them how they will log in...
and some initial task will perform. You can show a small piece of development and use it as a springboard to talk about where the project is headed.

Clearly, you will be able to show more early deliverables around screen configuration work than you will integration or conversion work. Plan for that accordingly and make it happen. Not only does this allow you to get the organization excited, but hopefully the project team as well. These frequent minor victories pay dividends in the form of increased and sustained engagement. If you focus on only the end product, then team members are likely to become disconnected from how their piece fits into the overall evolution of the project. Imagine trying to read a 500-page book that is only one paragraph. Agile allows you to break up your project into logical “paragraphs,” which in turn become manageable, meaningful chunks of output.

- **Sprint Demos: Make Them Meaningful** – Getting buy-in early and often is the key to the project’s success. Showing small pieces of work in progress (it doesn’t even need to be fully complete) allows the organization to take ownership sooner rather than later. Allowing for the organization to understand the evolution of the development process supports this early ownership initiative. It can also drive home the appreciation for the complexity of the project. Having demos at the end of each sprint helps to keep the organization engaged throughout the life of the project. The audience for these demos will certainly have an understanding of the day-to-day work on the project. Allowing them to see a demo of new work and functionality every month or so fosters a continually renewed sense of engagement. Don’t be afraid to show them re-work as well. Let them see how the process drives toward a more complete product. You will no doubt discover opportunities for improvement as a result of feedback arising directly out of the demos.

The demos are an ideal opportunity to continually reinforce the overall goal of the transformation. While replacement of the core system is the cornerstone of the framework of change, don’t let implementation of the system itself be perceived as the end game. Tying key pieces of demonstrated functionality back to key initiatives that are part of the change strategy will enhance engagement within the organization.

Another key benefit is allowing organization’s leadership the opportunity to communicate early in development that the project is off track or going in an unintended direction. You don’t want to pull back the curtain at the end of the project to show off the shiny red car that you built – only to find out that senior leadership was expecting a yellow one. Even if the yellow one is far better for them than the red one, it’s much harder to communicate that at the end. Allow them the opportunity to buy into the “yellow” concept (or perhaps come up with it on their own) before you even buy the paint.

It may be that leadership did not have a clearly articulated vision from inception or the vision has changed since then. It may be that the SME’s that have been provided to the project don’t have a picture of the full vision or don’t agree with it. Whatever the reason, you want to know this sooner rather than later. It is better to identify this early, get to the bottom of it, and solve it. Even if this causes a delay in the project, you should be able to push the drivers and reasons clearly into the spotlight. Better to be delayed for a month or two with full transparency than to deliver an end product that is perceived as a “close, but no cigar” or a complete failure.

- **Prioritizing Requirements** – It is inevitable that new requirements are introduced during the development phase of the project, especially using Agile methodology in which the organization is learning the product and seeing progress from sprint demonstrations as they move through development. It is important for the development team to help the business distinguish between the “must-haves” and “nice-to-haves.” The “nice-to-haves” can be revisited at the end of the development phase if there are sufficient resources available to work on them without jeopardizing the project deadlines or the ability to upgrade in the future. Otherwise, they can be scheduled for future releases.

The developers should be open to challenging those requirements that pose risk and potential inefficiency. They should use their skills and experience to help avoid these pitfalls and requirement gaps. As the product
stewards, the developers are responsible for steering the organization toward making the correct decisions in prioritizing requirements.

Sometimes, because of time and resource constraints, the “must-haves” necessitate prioritization as well. If there is a work-around available to complete the claim, billing, or policy lifecycle that can replace these requirements, they should have the lowest priority and be developed at the end of the project if the time and resources permit. These requirements may have to be scheduled for future releases if they make logical sense.

What happens if one or more of the business owners leave the organization or take on a new role? This can be a significant threat to the project. The new business owner may not agree with the direction or the previously agreed upon prioritization. Take the opportunity to walk the new business owner through the current plan and the rationale behind it as quickly as possible. Try to build early consensus for the plan that is currently underway. If that is met with resistance and a clear change in direction appears imminent, be sure to carefully follow the project’s governance model—particularly if such changes will result in increasing the scope, timeline, or budget. The steering committee must always be the deciding factor in such issues. Even if the new direction is the correct one, the communication must be inclusive and decisions made by the right resources with the right authority.

**Off-Shore Challenges**

- Teams are most effective when they are co-located.
- Offshore teams may not have the same business support.
- Time Zone differences may make work/life balance more difficult given the need for meetings held “off hours.”
- Language differences may create additional challenges in iteration management.
- A product owner for multiple distributed teams is difficult to maintain.

**Off-Shore Agile Lessons Learned**

- **Code Quality**
  - Intense management is needed to have sufficient rigor in code quality and architecture decisions requiring solution architects to be co-located with the offshore team.
- **Communication/Collaboration**
  - Nothing replaces face-to-face interactions. Schedule a face-to-face meeting among the teams early in the project.
  - Rotate key Business Analysts offshore (3 weeks off-shore/3 weeks on) to function as business proxies and address with the challenges in remote collaboration between developers and BAs.
- **Facilities/Infrastructure**
  - Issues with ISP stability and network latency can exacerbate the remote communication challenges.

**Using Offshore Resources** — After scaling onshore Agile teams successfully, organizations can begin to consider an offshore model. The key to realizing the value of offshore Agile is implementing just enough structure to connect teams with the rest of the onshore organization while still allowing them to be Agile. Offshore resources can help you fill out all of the roles within your team. Costs can also be effectively managed by leveraging these resources.

A deliberate approach must be employed to keep these team members fully engaged with the rest of the team and the project. Consider using partially offset hours to make sure that their schedules overlap somewhat with that of the onshore team. Try to have at least some of your scrum meetings during this overlap. Partially overlapping schedules help to reduce isolation, while still providing the off-shore resources a reasonable work-life balance. These schedules also allow for the opportunity for development work to continue essentially “after hours.” Requirements that are completed during the day can often be developed overnight by the offshore team.

If your offshore team is large enough, you may consider devoting a lead to managing the group. This lead can not only help to facilitate scheduling and other logistic challenges, but can also serve as the champion of this
staffing strategy. The lead can also help to smooth any bumps along the way that may be caused by cultural or communication challenges.

- **Visibility and Accountability** – Agile not only breaks the project into manageable pieces, but those pieces are logical milestones against which to measure progress. The end of the sprint (and subsequent demo of the developed functionality) clearly sets interim deadlines within the project timeline. Ongoing communication within the project team calls for team members to be accountable to each other—not necessarily to the project leadership—for results.

Daily stand-up meetings will become the pulse of the project. Each team member must deliver or the results will not be met. Stumbling blocks can be identified, defined, contained, and solved in an open and deliberate fashion. Don’t be afraid to call someone out for either over-committing or under-committing. Each of these poses unique issues. You shouldn’t assume that this equates to each person on the team completing the same number of units of work in a given time period. Make sure that the work completed lives up to each individual’s potential. Variances (within a reasonable tolerance) are natural and should be expected.

As the team updates happen on a daily basis, it is also important for the project’s leadership to have ongoing check-ins as a team and also with the organization’s executive sponsor. Regularly scheduled leadership meetings and executive team reviews help keep everyone in step. Dashboards can be a valuable tool here to track the project’s progress against plan. Don’t over-reach here. This is the proper forum for candid discussions about how the project is advancing. If a stumbling block has been identified, succinctly define it, explain the risks, and communicate the plan to mitigate those risks.

The regular rhythm of the project keeps an ongoing momentum. While the velocity of the project will certainly have its peaks and valleys, it becomes difficult for the project to stall for more than a few days. Clearly delivering new functionality at each sprint all but eliminates the question: “What did the team really accomplish last month?”

- **Aligning Data Conversion to the Overall Project** – If your project is to include converting legacy data into the new solution, a deliberate approach must be taken to properly fit that portion of the project within the Agile framework. While the data conversion effort itself clearly does not lend itself to Agile, it does have to work in harmony with the rest of the project. The conversion effort must be planned for in parallel with the rest of the development effort. Keeping everyone current with the latest updates to an ever-changing data model will reduce re-work. Consider holding regularly scheduled meetings to discuss data model changes. These meetings should include at least some representation from every workstream, including conversion.

The conversion effort should be expected to contribute regularly to requirements. As these requirements are discovered through the conversion process, careful evaluation must be made to determine whether such changes should be confined only to the converted data or whether they should be applied unilaterally. If a change needed for the conversion effort will satisfy the business need going forward, there is no need for the system to function one way for converted data and another way for native data. Conversely, you should carefully guard against a “lowest common denominator” approach. The needs of conversion cannot artificially restrict needed functionality in the new system.
Looking Ahead

The ongoing transparency will help to solidify the perception of the success of the project. Even if the project evolved in a somewhat different direction from the original plans, business owners don’t have to ask the questions: “How did we get here?” or “Why did this take so long?” If the project had to be delayed, everyone has an appreciation for how the timeline evolved.

Ownership is key. If business owners were involved throughout the process, they are more inclined to discuss what “we” built vs. what “they” built. People will usually describe what “we” built in more positive terms than what “they” built. Proper use of Agile can help to move more into the “stuff we built” category. Allow the product owner to take pride in this stage of the finished product, while setting the stage with the organization for the next phase of the project.

If your team does not have experience with the packaged solution, it is critical to get help from those with experience. Resources that have experience in delivering the solution will certainly bring to bear insight into the complexity of components and potential obstacles. The road on which you are embarking is likely not uncharted territory—don’t go it alone.

While it is tempting to view delivery as the end of the project, it is nothing more than a stop along the way. Planning (and perhaps even some execution) for enhancements, future releases, and upgrades should already be underway. The ability to make upgrades over time was certainly one of the reasons that the organization selected a package solution. The stage should be set for future work along a continuous cycle of improvement.

Experience is the best teacher for fully optimizing an Agile implementation process. PwC has led client organizations through hundreds of projects utilizing Agile, many of which were the organization’s first exposure to Agile. PwC’s team is able to leverage this experience through a full end-to-end approach, starting with initial training of the team, all the way through deployment. Employing a solid combination of knowledge, tools, and experience is what allows organizations to leverage the principles of Agile in a way that supports the needs of the business, with an eye to efficiently managing not only the current project, but providing a solid foundation for driving Agile throughout the organization.
To discuss Agile methodology in more detail, please contact:

Christine Goldberg  
Principal, Advisory Services  
+1 440 315 3969  
christine.l.goldberg@us.pwc.com

Mike Mariani  
Principal, Advisory Services  
+1 602 364 8486  
michael.j.mariani@us.pwc.com

Ricky Raisinghani  
Director, Advisory Services  
+1 312 298 2695  
ricky.raisinghani@us.pwc.com

Douglas Honaker  
Manager  
+1 214 754 7566  
douglas.honaker@us.pwc.com