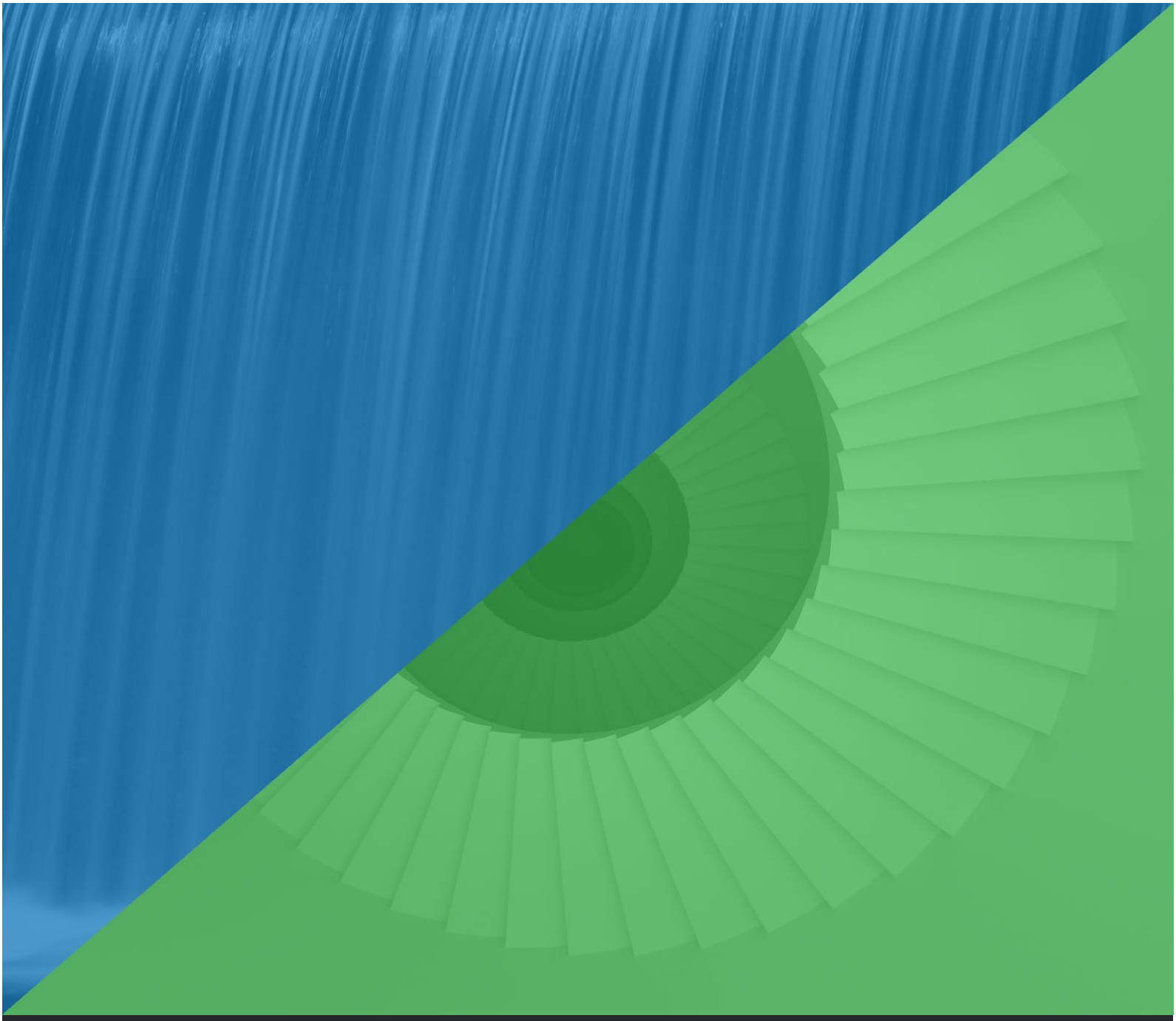


WHITE PAPER

GET MORE WORK DONE: A MANAGER'S GUIDE TO MIXING AGILE AND WATERFALL

Project management is shifting. Teams that were siloed before are now asked to work together to ease reporting pains and increase visibility across all types of work. Choosing Agile or Waterfall is becoming less of a choice as more organizations require a mix of the two. Solve the pain of this new work management style and learn how to stay productive by mixing methodologies.

AtTask

Executive Summary

Today's project leaders are faced with a problem: While most of their work follows a traditional Waterfall methodology, they're increasingly asked to collaborate with Agile software development teams. Agile and Waterfall lie at the farthest ends of the spectrum and conjure up deep-rooted opinions when teams are asked to pick a side, work with one another, or in some cases, choose Waterfall to complete software development projects.

> *This white paper will dive into the differences between the Agile and Waterfall methodologies in relation to software development, the pros and cons of each, best use cases, and finally, the roadmap for mixing the two.*

While these methodologies may be opposite in almost every way, the need for both is obvious, especially for development teams or PMOs that still have the responsibility of reporting up the chain to management. Waterfall reports aren't going away, and Agile development is the methodology of choice for many IT groups.

This white paper will take a deeper dive into the differences between the two methodologies. While both can be used to manage a wide range of project types, this paper will focus primarily on the application of the two methodologies to the world of software development, the pros and cons of each, best use cases, and finally, the roadmap for mixing the two. Solve the pain of attempting to mix two very different methodologies and learn how Agile and Waterfall can help you get more work done, deliver better metrics to executives, and create the right environment for all types of work.

WATERFALL

The Waterfall methodology is the most common project management approach in today's workplace. This top-down, classic approach works well in a number of areas—construction, manufacturing, repeatable services, and any process that produces a tangible product. These projects thrive with a methodology that sees progress flowing downward, just like a waterfall.

The core of the Waterfall methodology is a detailed plan that is scheduled and executed sequentially to a final resolution. Waterfall is concerned with deadlines and recordkeeping. It moves from one step to the next with no room for deviation. If one step is held up, the costs and time added to the project can derail the success of the entire operation. Waterfall is built on dependencies—step A must be complete before step B can begin. This works especially well in a top-down culture, in industries that have few competitors, with established businesses, and in any process related to compliance or regulation.

SUCCESSFUL WATERFALL CONDITIONS

- Physical Product Production
- Top-down Culture
- Few Competitors
- Established Business
- Repeatable Process
- Regulation or Compliance Focus

THE HISTORY OF WATERFALL

Originating in the industries that follow a specific step-by-step pattern—manufacturing and construction—Waterfall was adapted for software development in the the 1950s. The formalized description was published in 1970 by Dr. Winston Royce in a paper titled "Managing the Development of Large Software Systems."[1]

This placeholder methodology was described by Royce as “risky and [inviting] failure.” It was only intended to work for software development after substantial testing and a clear understanding of customer needs.

THE WATERFALL MODEL

In today’s work environment, IT groups can be asked to follow this methodology in order to match the rest of the organization. In a Waterfall approach for IT, software developers must follow eight specific steps.[2] As each phase ends, it triggers the opportunity for the next phase to begin.

> *“Waterfall was only intended for software development after a clear understanding of customer needs.”*

The Eight Phases of Waterfall

1. Conception
2. Initiation
3. Analysis
4. Design
5. Construction
6. Testing
7. Production/Implementation
8. Maintenance

WHEN TO USE WATERFALL

The Waterfall methodology is best suited for projects that have a clear picture of the final product. It only works if clients don’t expect the opportunity to change the scope of the project after it has commenced and when the definition of the project, that detailed plan, is more important than speed. Code is delivered to the client when it’s complete, not as it’s generated.

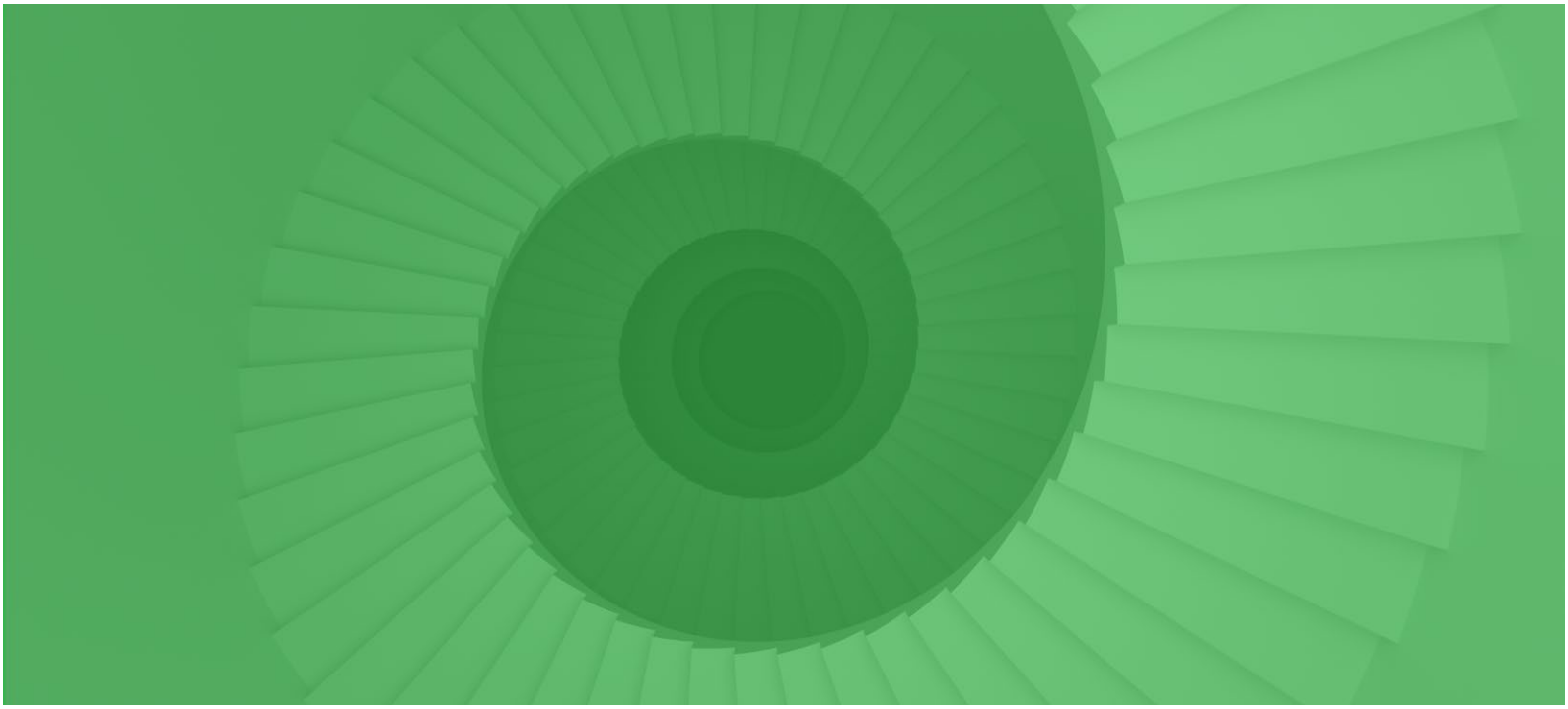
For the rest of the organization, Waterfall is used for traditional project management. The strict, deadline-driven methodology plays well for projects that have a clear vision of the end result and are dependent on completion of specific tasks before the next tasks can begin.

WATERFALL PROJECTS

- Step-by-step Process
- Clear Milestones
- Recordkeeping

THE PROS AND CONS OF WATERFALL

PROS	CONS
Incredible Detail Planning, planning, and more planning—the number one advantage of Waterfall. This meticulous preparation results in a very detailed project plan that should, according to the intentions of Royce, be based on a clear understanding of customer needs if it's used for software development. Those that follow the Waterfall methodology keep scrupulous notes, and future projects benefit from this historical detail.	**Don't Think About Going Back** Waterfall follows a strict blueprint, and any type of departure from the original plan is difficult, at best. If the initial requirements are faulty in any way or the client's needs change during any phase of the project, the team must begin again and start with new code. In these cases, the project is often scrapped and a new project begins, adding unplanned dollars and hours to the bottom line.
You Get What You Ask For At no point are clients unclear about what will be delivered in a software development project that follows the Waterfall methodology. The project size, cost, and timeline are clearly outlined and delivered to the customer before the first line of code is written. Waterfall doesn't play nicely with deviation from the project plan; therefore, the project doesn't vary from the already clearly defined path set at the beginning. If the client asks for X, the development team will deliver X, not Y.	**Wait Until the End for Testing** Testing code in a Waterfall project is only completed at the very end. As code is completed, it's set aside for final QA, a phase that is planned to take significant time to fix the inevitable bugs that are found during testing.
Doesn't Skip a Beat The recordkeeping and crystal clear expectations leave no room for misunderstanding. If the development team somehow experiences any turnover, the project hand-off is seamless and easy. The outlined plan and documentation from step to step allow a new member of the team to step in and contribute without derailing the timeline of the project.	**No Room for Evolving Needs** The client's needs generally evolve after the project is underway. With a Waterfall project, those emerging needs cannot be addressed. The client must clearly communicate all of the anticipated needs that the project will address before the project is started. The strict, step-by-step process simply doesn't allow for changing requirements.



AGILE

The Agile methodology offers a more granular, responsive approach to project management, commonly used in software development. Unlike Waterfall, Agile is reactive. It allows workers to self-organize, self-prioritize, and decide what tasks are most important from the queue of the project plan. Agile allows for the delivery of small, working sections of the overall project to ensure the customer's needs are constantly considered throughout the process. Quick feedback is valued, and the discovery of new requirements allows developers to tailor future code in order to deliver an improved product.

The idea of failing fast is prevalent in the Agile methodology. Testing is constant and bugs are fixed along the way instead of waiting for one comprehensive testing cycle at the end. Projects that align well with Agile include those in a bottom-up culture, projects related to a fast-moving space or industry, and software or services that have a lot of competition and need to adjust quickly to meet market requirements. Agile maximizes creativity in situations that feature volatility in requirements.

It's a bottom-up, pull-based approach and requires a similar culture to succeed. Agile is a complete shift from Waterfall and focuses on a new way to work with employees, treat customers, and gain value from projects.

SUCCESSFUL AGILE CONDITIONS

- Software
- Services
- Bottom-up Culture
- Highly Competitive Industries
- Fast-moving Space

THE HISTORY OF AGILE

"The Manifesto for Agile Software Development" was published in February of 2001 after 17 software developers met at a ski resort in Utah to start a dialog about development methods.

The manifesto defines four values that the Agile methodology promotes: individuals and interactions, working software, customer collaboration, and responding to change. In short, the Agile methodology is not concerned about a planned project that must be completed in a sequential order. Agile ultimately puts the customer and the software over any type of plan, contract, document, or process.

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions	over	**Processes and tools**
Working software	over	**Comprehensive documentation**
Customer collaboration	over	**Contract negotiation**
Responding to change	over	**Following a plan**

That is, while there is value in the items on the right, we value the items on the left more.[3]

THE AGILE MODEL

Encouraging rapid progress, the Agile methodology promotes iterative development, delivering pieces of a whole to ensure the customer's needs are met along the entire project path. Project requirements change along the way, and the methodology allows for agility in dealing with those unpredictable conditions. In some cases, Agile can be the methodology of choice for other areas of the organization like marketing or a creative services team. The agility and frequent delivery of content to the customer allows the team to get constant feedback for a higher-quality product at the end.

Agile values quality and customer input over contracts and predictive steps. The methodology breaks large tasks into small tasks and sections each into specific time frames. The typical timebox for an Agile team (known as a sprint or iteration) spans one to four weeks. Because each task is broken into bite-sized pieces, each member of a cross-functional team can contribute in parallel—and at the end of the timebox, the team delivers a working product that can be evaluated by the customer. Everything from coding to testing is completed within the time frame to ensure the client has a working piece of software at the end.

COMMON AGILE TERMS

Burndown Chart

A visual representation of the remaining work over a period of time. The burndown chart measures progress at a project level and an iteration level.[3]

Cross-functional Teams

Teams of four to eight individuals who, as a whole, make up the resources required to deliver a complete, working product.

Daily Standup/Scrum

Short, daily meetings where team members individually report on what was completed yesterday, what needs to be completed today, and any possible hurdles that could prevent work from progressing.

Epic

User story with a sizeable list of requirements. Epics are generally pieced into smaller stories for easier planning.

Product Backlog

The collection of user stories or requirements that can feed future development. This backlog is prioritized so that the most important issues are addressed in near-term releases.

Scrum Master

Not considered the leader of the team, the Scrum Master is the buffer between any incoming requests and distractions and the team. The priority of the Scrum Master is to keep the team on task.

Sprint/Iteration

A boxed set of time, typically one to four weeks. Working product is delivered at the end of every sprint/iteration.

Sprint Backlog

List of work that the team plans to complete before the end of the sprint.

Story Points

The units used to measure the difficulty or complexity of each task within an Agile project. The more story points assigned to a task, the longer that task will take to complete.

User Story

A one or two sentence description of what the end user needs to see. User stories are used to estimate the time needed to build the product. [4]

THE 12 PRINCIPLES OF AGILE

1. Satisfy customers through continuous software delivery
2. Welcome changing requirements and needs
3. Deliver working software frequently
4. Encourage daily communication between business people and developers
5. Trust individuals to get the work done
6. Communicate as a team in face-to-face conversation
7. Measure progress by working software
8. Maintain a constant development pace
9. Emphasize good design practices and technical excellence
10. Keep it simple
11. Allow teams to self-organize for better productivity
12. Adapt the process as needed[5]

POPULAR AGILE METHODS

Scrum

Offering a flexible environment for software development, Scrum is one of the most popular Agile methods. It's iterative, incremental, and relies heavily on frequent team communication and interactions with the customer. Work can be moved in and out of the sprint as needed to meet the customer's evolving needs.

Kanban

The Kanban method focuses on just-in-time delivery. A collective queue houses tasks, and workers pull work into the Kanban board. This method isn't time bound, but the vertical lanes do have WIP (work in progress) limits—limits on how much work can be in one section of the board at a time. Work is moved through columns generally labeled Ready to Get Started, New (or In Progress), The Development Cycle (often different for each team), and Complete.

eXtreme Programming (XP)

XP features frequent releases in "short development cycles."[6] Think of this method as *strict scrum*. Work isn't moved in and out of a sprint, whereas in Scrum, it's a more flexible, fluid approach. Features are broken into granular tasks and functional code is released quickly.

Lean Software Development

Similar to Lean Manufacturing, Lean Software Development focuses on eliminating any possible waste from the development process and delivery of code. Everything from unnecessary functionality to slow communication between teams is regarded as waste and avoided to produce a better product. Team members are empowered to stop the process at any time if any functionality isn't meeting quality requirements. Workers can insist that any issue or bug be fixed before the process moves forward.

AGILE METHODS

- Scrum
- Kanban
- eXtreme Programming (XP)
- Lean Software Development
- Feature-driven Development (FDD)
- Adaptive Software Development/ Crystal
- Dynamic Systems Development Method (DSDM)
- Crystal Methods
- Disciplined Agile Delivery

WHEN TO USE AGILE

There are three keys to a successful Agile project: 1) rapid production must be more important than quality, 2) clients will need the opportunity to change the requirements of the end product, and 3) skilled and independent developers are able to work on the project.

If the definition of the end result is more important than putting new code in front of a customer frequently during the project, or if the team is simply not able to work in a timebox, the project will likely fail using an Agile methodology.

THE PROS AND CONS OF AGILE

PROS	CONS
Revisit, Revisit, Revisit The Agile methodology allows developers to revisit specific steps in the process as many times as necessary. As the scope of the project changes, developers can rewrite what needs to be done in order to meet those changing needs. Unlike Waterfall, this allows the customer to have a continual dialogue with the development team to increase the chances of an improved product at the end of the project.	**Not Much of a Starting Point** Unlike Waterfall, the Agile methodology doesn't begin a project with a meticulously scheduled project plan. The planning is done in parallel with the actual work. While this allows for quick transition, it doesn't offer much confidence to customers who don't trust the development team or who need a specific product on a tight deadline.
Get it Out, Get it Tested The overarching project is broken into manageable tasks, allowing the development team to focus on getting code ready and testing that code within the same timebox. Frequent testing delivers a higher-quality product at the end and allows for faster delivery than Waterfall, which requires a single, time-consuming test pass at the end of the project.	**End in a Different Place** The beginning and end of an Agile project may look very different. A project that starts down path X may very well end up on path Y because the project requirements were changed as new code was delivered and tested. While this works for projects that need agility, it may not sit well with customers and can cause problems for other areas of the organization. For example, a new release may not be defined, which makes it hard for marketing to communicate new features to customers.
Complete Customer Focus Agile centers around the needs of the customer. By utilizing acceptance criteria within each user story, the team understands what the customer wants to see. And, with the ability to deliver code in a more frequent manner than a Waterfall project, Agile allows developers to fail fast and quickly change the direction of the project without losing too many hours or dollars.	**Heavy Reliance on Others** The Agile Manifesto and the principles on which it is based rely heavily on good, trustworthy team members. A poor project manager or developer could result in work that's not completed or a series of code sprints that put the project over time or over budget.

MIXING METHODOLOGIES

Traditionally, when Agile and Waterfall are compared side-by-side, they don't appear to have similar goals. Mixing the two methodologies can be tricky, and most businesses that attempt the mix without due diligence end up experiencing a few common pitfalls.

Waterfall is a predictive methodology. Step A comes before step B, and so on. Features of a product are guaranteed, and there is no feasible way to revisit a step in the process. Any holdup during the project will create a late delivery.

Agile, on the other hand, is reactive. The development team can't promise delivery of a specific, outlined, pre-planned product, but they can promise delivery of working software on a specific date. Because they cut the project down into smaller pieces, they can produce sections of working code at the end of each sprint. This timeboxing method ensures that the customer will see functional software continually throughout the project.

COMMON PITFALLS

1. Zero Training
2. Zero Buy-in
3. Zero Translation

COMMON PITFALLS

1. Zero Training

Failing to properly train the group that should adopt the Agile methodology is the first common pitfall. Often, a company or group will decide to employ this new methodology without any instruction on how to assign stories, plan sprints, etc.

Project leaders assigned to manage these teams are also often not trained on how to wrap iterations into an overarching Waterfall plan.

2. Zero Buy-in

Culture is the key to success in implementing an Agile strategy for development. Without middle management's buy-in, the Agile team will be siloed, forgotten, and probably disbanded at some point. Managers are responsible for showcasing the value of their teams. Without their help, an Agile team will flounder in a world familiar to due dates, thorough planning, and task dependencies.

3. Zero Translation

The metrics that each methodology produces don't automatically translate. A timebox doesn't equal a due date and a story point doesn't always equal one hour or one day of work in a Waterfall plan. Companies that don't anticipate the differences and make a plan for metric translation will most likely scrap the Agile approach, returning to the familiar, rigid arms of a Waterfall methodology.

MIXING AGILE AND WATERFALL

Agile and Waterfall need to exist in the same space. Both bring great benefits to the table and work for a variety of projects. For example, many technology companies will run their infrastructure projects (standing up servers or building other physical environments) using Waterfall, and build their software applications using Agile. Likewise, many application development teams prefer Agile methodologies, but executives need to make high-level plans using more traditional, Waterfall measures.

The bottom line is that in today's world, we still need the rigidity of a Waterfall project, but we also need the responsiveness of an Agile approach. It's important to remember that communication is key when mixing the two in the same environment. Managing expectations between the two groups or two sides of the business will allow for positive interactions.

Culture

Implementing the Agile methodology in any workplace is difficult if the culture doesn't align with the principles of Agile development. Middle management is key to the success and adoption of Agile. Managers need to be trained and should fully understand the value and advantage of Agile methodologies. They need to clearly see the translation between Agile jargon and Waterfall definitions. This is especially important for reporting purposes and in proving and measuring the value of the Agile team in a culture that is more familiar with Waterfall.

Metrics

Allow Agile teams to function as Agile teams, but report to the rest of the company in terms that traditional project leaders want to see. Metrics should be delivered in Waterfall so the Agile team can be compared, side-by-side with other teams. Project process should be translated into the language that the C-suite understands. Is the project on time? Is the project on track to deliver the requirements? Is the project within budget?

Translating metrics takes some effort, and some compromise will be necessary. It's important to focus the data into three main areas: what is the team working on, what are they doing right now, and when will they be done? These metrics transcend methodologies. Story point estimates can then become planned hours in a traditional project management report. Iterations or sprint dates can become task constraints. The manager simply has to understand the Agile team's definition.

Benchmarks

In order to determine if an Agile team is effective in a Waterfall environment, key benchmarks must be outlined from the get-go. Have you acquired more customers because new features have been added to a product? Does a feature in an A/B test drive sales numbers? Define the benchmarks, collect the data, and assess the value of the Agile team. Waterfall metrics should also be gathered for quick and easy comparison.

In the end, it's about avoiding the overall fear of letting go. Many IT groups or PMOs resist the adoption of Agile or the integration of Agile into their process because they fear they will lose control. When projects that start in Waterfall are turned over to the Agile team, those projects are now Agile projects. Work inside that methodology is prioritized according to the availability of resources that use Agile. A task that lives in an iteration cannot be locked down by dependencies while it's in Agile; otherwise, it's just another Waterfall task. A well-organized Agile team will deliver products on time.

The takeaway: let Waterfall teams be Waterfall and let Agile teams be Agile. Don't combine the methods, let the two work side by side, understanding the needs of the other while staying true to their own. It's the secret to making your teams more productive.

TOOLS TO HELP

Incorporating an Agile team into a Waterfall environment can be made easier with the help of the right technology. A solution that serves both masters will promote success for both Agile and Waterfall teams.

When evaluating technology choices, a number of factors should be considered, but data integration and reporting are paramount. Find a solution that automatically translates the data from your Agile team into Waterfall reports.

The right technology will roll up the Agile team's data into Waterfall-like dashboards, reporting on the progress of all teams, all work, and all methodologies. When a common set of metrics are in place under one reporting umbrella, facilitating good communication and connecting the two sides of the business becomes easier.

BENEFITS OF MIXING METHODOLOGIES

Mixing methodologies isn't about pigeonholing one methodology into another. It's also not about forcing one group to adopt a new solution that won't work for their projects. Agile and Waterfall are the right options for the right work. Mixing the two in the same organization should benefit each group and the business as a whole. Connecting the two methodologies must give Agile and Waterfall the same weight; and implemented correctly, using both will increase visibility and productivity.

> *"In order to determine if an Agile team is effective in a Waterfall environment, key benchmarks must be outlined from the get-go."*

Increased Visibility

With proper planning, training, and benchmarks, an organization that allows both Agile and Waterfall methodologies will have complete visibility into what work is in the queue, what work is in progress, and what work will be completed in the future.

Increased Productivity

Allowing teams to work in their methodology of choice opens doors to increased productivity. Software teams are more inclined to pick an Agile methodology to increase production of working code. Project managers and other departments may choose Waterfall because it's simply a better approach for projects that require strict deadlines and have a long list of dependencies.

CONCLUSION

You don't have to continue down the path of disjointed methodologies that distance departments or teams and create chaos with reporting. Both methodologies can work together and with the right technology, you can prove the value of all work to your organization.

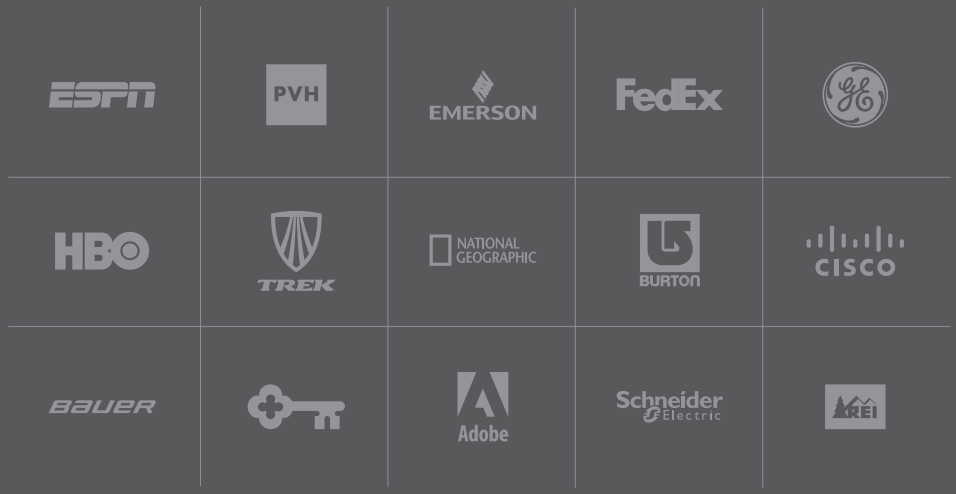
Mixing methodologies is made easy with the help of a tool that offers a clear line of sight into both Agile work and Waterfall projects.

Meet AtTask

AtTask is a cloud-based Enterprise Work Management solution that helps IT departments, PMOs, and other enterprise teams work the way they want while reporting in the same manner up the chain. AtTask offers a complete, adoptable solution—powerful enough for technical users, intuitive enough for business stakeholders, and flexible enough to support Agile, Waterfall, or a mix of the two. It works in the same ways you do.

To learn more about AtTask Enterprise Work Management for IT project management, and how it can help you mix methodologies and increase work visibility, please contact us at the following:

www.attask.com + 1.866.441.0001 + 44 (0)845 5083771



"AtTask gives us one system to track and manage all of the projects our team has underway. We introduced a new Phase Gate methodology, with Agile design embedded in it, and we needed a solution that would get adopted while helping us enforce corporate software development standards. AtTask offers intuitive reporting and a platform ideal for mixing methodologies."

Emerson, Commercial and Residential Solutions

Endnotes

1. Royce, Dr. Winston W. “Managing the Development of Large Software Systems.” http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf (accessed July 3, 2014).

2. Wikimedia Foundation. “Waterfall model.” Wikipedia. http://en.wikipedia.org/wiki/Waterfall_model (accessed July 3, 2014).

3. Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. “Manifesto for Agile Software Development.” Manifesto for Agile Software Development. http://agilemanifesto.org/ (accessed July 3, 2014).

4. “Agile Terms Glossary.” http://www.telerik.com/teampulse/agile-vocabulary (accessed July 3, 2014).

5. Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. “Manifesto for Agile Software Development.” Manifesto for Agile Software Development. http://agilemanifesto.org/principles.html (accessed July 3, 2014).

6. Wikimedia Foundation. “Extreme programming.” Wikipedia. http://en.wikipedia.org/wiki/Extreme_programming (accessed July 8, 2014).