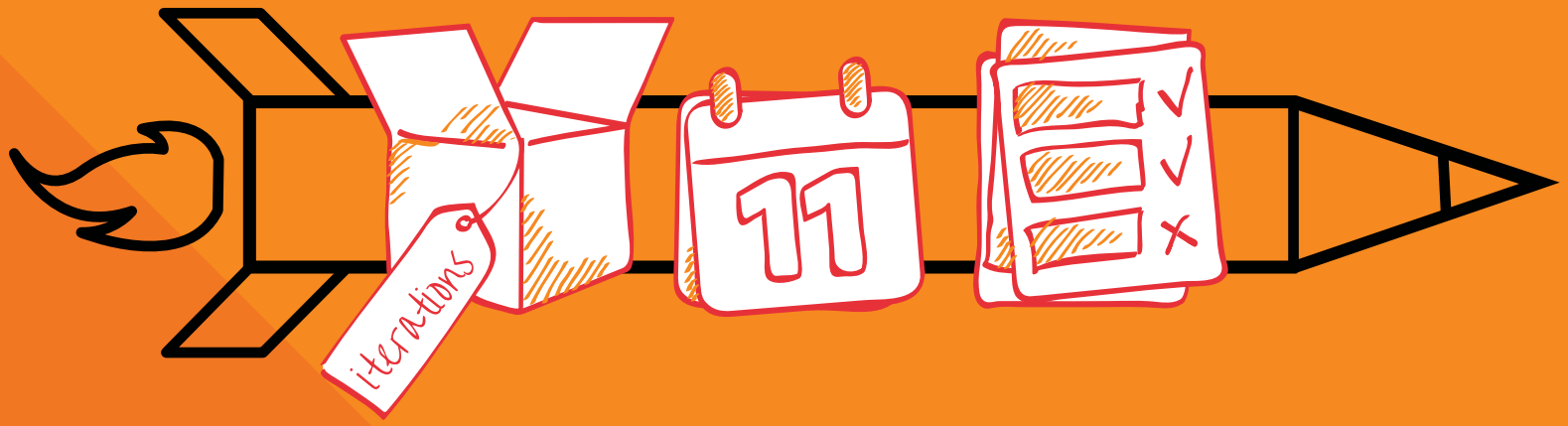
Telerik TeamPulse

THE ULTIMATE AGILE PLANNING HANDBOOK



CHAPTERS

Chapter:

ELEMENTS OF AGILE PLANNING

Perhaps some of the most common questions asked about Agile revolve around planning. Of course, some consider planning to be incompatible with Agile principles. In practice, this is just not the case. Agile practitioners find that they plan as much or even more than their non-Agile counterparts. Agile doesn't discourage planning, instead it promotes continual planning, a process some call "continual re-planning." This is because Agile places more emphasis on providing value than on following a static plan.

In contrast, a traditional project places emphasis on detailed planning done prior to investments in time and resources. This "big upfront plan" attempts to plot the course of the entire project, trying to account for the complete list of anticipated requirements for all phases of development. Teams are then asked to execute against the plan and diligently track any and all plan divergences. However, plan divergence is discouraged because of the significant time and effort placed into the upfront planning. Success in traditional projects is largely a reflection of how well the team adheres to the original plan.The simple goal of an Agile team is to maximize customer value and to do so as efficiently as possible. Agile teams embrace and expect the fuzzy view of customer requirements pre-project, recognizing that the true value emerges as customers begin to consume the incremental components they receive. On an Agile project, teams are encouraged to plan the project incrementally so that time typically invested into "big up front plans" is instead redirected to responding to the feedback the team receives as value is discovered along the way. That does not mean that Agile projects work without time and budget constraints. In fact, quite the opposite is true. Agile projects flourish with time and budget constraints as these help ensure customers scrutinize and prioritize value they truly need.

This paper provides guidance about how to make planning more "Agile."

Chapter:

WHOLE TEAM APPROACH

Unlike traditionally run projects which are planned by an "expert," Agile planning is done with the entire team. Agile encourages group participation in all forms of planning and estimation. Instead of one person following a linear process of assessing requirements and project constraints, then producing a plan designed to drive schedule, budget, and resources to deliver the end project, the whole team is engaged.

Including the entire team in the Agile process ensures communication between team members, and more importantly, with users of the software they are building. Here collaboration is key, because over-reliance on documents as the primary communication vehicle leaves a great deal of information lost in translation. The most effective way for the team to truly understand expectations is to work together to understand the problem, then form a resulting plan. Team planning sessions focus on communication and collaboration, and this added communication leads to a much better plan. Team members have the opportunity to jointly understand each requirement they are asked to build. They then work together to determine the amount and kind of work required to deliver the requirement to a customer.

More importantly, when the team plans together they can take into account the realities of their shared experiences. Agile promotes the tenet that the people doing the work should be the ones producing the plans as they have the best knowledge about how the work will get done. After a collaborative planning session, each team member is completely aligned with the problem they are trying to solve and the shared approach to the solution.

Chapter:

TIMEBOX – EVERYTHING

Timeboxing refers to the act of putting strict time boundaries around an action or activity. For example, you may want to timebox a meeting to be 30 minutes long to help ensure that the meeting will begin and end on time with no exceptions. When you timebox an event the result is a natural tendency to focus on the most important "stuff" first. If you time box a meeting, you would expect then that the absolutely required content of the meeting be completed before the end of the meeting. Other meeting agenda items that are not as important may get moved to another meeting. In essence, timeboxing is a constraint used by teams to help focus on value.

One important timebox that Agile promotes is the project itself. Contrary to Agile mythology, Agile teams prefer to have a timeboxed project since it offers a fixed schedule and a fixed team size. With these project constraints the team can better work with customers to maintain a laser focus on value, which ensures the team is building and delivering the most valuable work as soon as possible—leaving the less critical tasks to the end. Timeboxed projects may mean that some requirements won't get implemented. However, it will help to ensure that what is developed is truly the most essential of the required features.

Timeboxing also helps to prevent a common problem in software development called "feature creep," where teams incrementally add features to software without scrutinizing relevance or need. Feature creep leads to wasted effort in both the development and maintenance of code and can significantly impact quality and timelines on a project. Timeboxed project teams work to minimize the effort and resources needed to achieve the expected value. Some refer to this as the minimum viable product or the minimum viable feature set. Timeboxing iterations places emphasis on ensuring that teams do not experience feature creep.

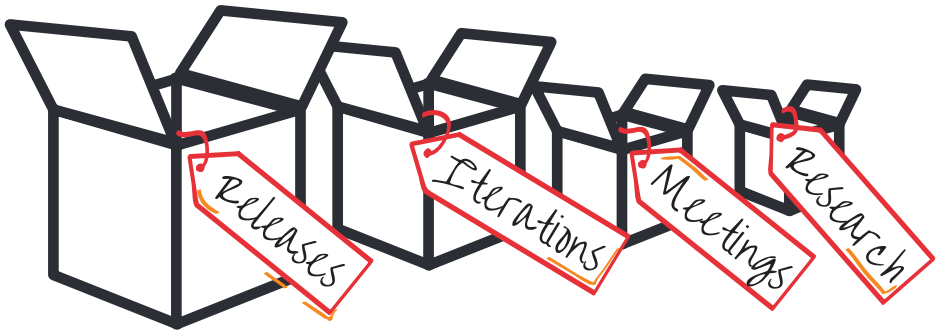


FIGURE 1 - ON AN AGILE PROJECT EVERYTHING IS GIVEN A TIME CONSTRAINT

Chapter:

ITERATIONS

Agile teams not only like to have timeboxed projects, they also prefer to break projects down into smaller timeboxed durations, commonly referred to as iterations. You may have heard of iterations by a different name if you are familiar with the Scrum agile methodology where they are referred to as Sprints. Iteration lengths are fixed and not flexible in anyway, meaning it will end regardless of whether all assigned work is completed. Outstanding work is either assigned to the next iteration, or reprioritized amongst the remaining tasks. The general rule is that iterations should not exceed 30 days in duration, although some teams use iterations that range from 4 weeks to a single day. The length of an iteration is largely determined by how a team and customer are required to work together.

Iterations are important to Agile teams as they represent the block of time during which they will produce the mostly finely delineated plan. Because the iteration is a shorter duration than the entire project, the team can finish an iteration plan in a few hours after which it can get deconstructed right down to the task level fairly accurately. At the beginning of an iteration, a team will work with the customer to select an appropriate amount of requirements to include, with the intent being that all chosen requirements will be completed within that iteration. The team will then work together to break-down the selected requirements into smaller pieces until the team is happy with the level of definition required to do the work. At the end of an iteration, the team releases some form of software for review by the customer or the customer advocate. For example, in the Scrum methodology, the Sprint culminates in a Sprint Review meeting with the customer to demonstrate the software and gather feedback about what was produced during that Sprint. If the team and customer want tighter feedback because requirements are rapidly emerging, then it's a good idea to schedule shorter duration iterations.

Having iterations that are too long (many months, for example) require much more planning and have a higher risk of slippage and error. In addition, the feedback loop between the customer and team is much longer providing a greater opportunity for the team experience rework due to errors or omissions. However, having iterations that are too short can also cause problems. With every iteration comes some overhead—specifically iteration planning—reviews and retrospective meetings. In addition, short iterations leave teams struggling to produce something of value before time expires.

Chapter:

ITERATIONS

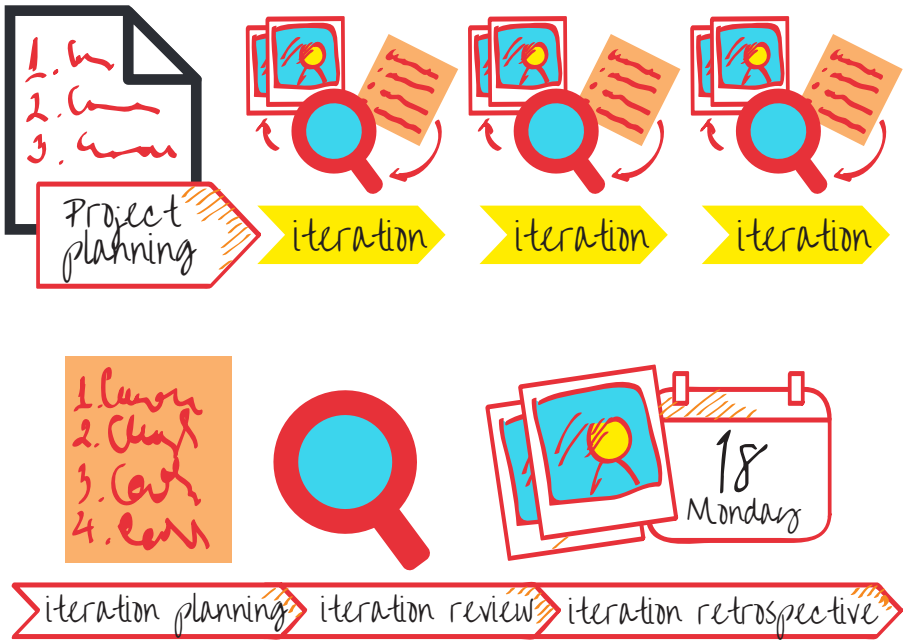


FIGURE 2 - PROJECTS ARE COMPRISED OF FIXED DURATION ITERATIONS.

Choosing the right iteration length is very much about finding the right balance between the team and the customer, so be ready for some trial and error. Two-week iterations are very common among Agile teams, so this is a good place to start for those new to the Agile development methodology.

Utilizing iterations also helps combat two very common and human conditions, often referred to as "The Student Syndrome" and "Parkinson's Law." The Student Syndrome refers to school students that wait until the very last minute to start their project assignments. Parkinson's Law refers to the occurrence whereby work assigned to teams—regardless of how simple—expands to fill the allotted time. Shorter iterations help combat both of these conditions and the associated risk and waste.

How the TeamPulse team does it - Planning

The Telerik TeamPulse team uses two-week sprints and aims to be "done done" by the end of every sprint. This means that every two weeks, the team literally has a completely shippable product—including tested functionality, documentation and installer packet.

Each sprint starts with a planning session, lasting from 90 minutes to 2 hours. Prior to the sprint planning session, there is a grooming session where a couple team members from each role sit together with the product owner and review the user stories for the next iteration. This way, we make sure the user stories we review during planning are as mature as possible.

During the planning session, each team gets aligned with the goals for the iteration, reviews the stories, conducts group estimation (using planning poker to get relative estimates), decomposes the stories into work (TeamPulse tasks) and produces the initial sprint plan. The team also works to provide the initial set of acceptance criteria for each story—something used to define what "done" means for the story.

At the end of each iteration planning session, we always run the TeamPulse Best Practices Analyzer (BPA) to see if the completed work conforms to development best practices. The BPA runs through the items planned for the iteration and checks for stories without estimates, tasks or acceptance criteria, as well as many other factors critical for the success of each iteration and project. It even points to those stories, so we can quickly go in and update them.

Chapter:

PRIORITIZED AND SIZED BACKLOG

For agile planning to work, you must have a prioritized and sized (estimated) backlog. A prioritized backlog is simply a list of work that needs to get done that is ordered by priority. Essentially, the work at the top of the list is the most important and should be done first. In many cases, customers have a difficult time prioritizing lists of features, however, this makes the prioritized backlog even more important to produce. Another requirement of this backlog is that each of the items must be sized, which is another way of saying that the item should have an estimate. The difference between sizing a backlog item and estimating it, however, is that Agile teams would much rather provide an estimate of the size of an item, using some arbitrary sizing scale than attempt to determine how many hours it will take them to complete the work.

Prioritizing a backlog should be a very simple exercise. Customers should be able to take any two items in the backlog and specify which of the two items is more important to them. The one with higher importance should be at the top of the backlog. This form of prioritization is called binary prioritization and can be used to sort the entire backlog from highest priority to lowest priority items. An alternative to binary prioritization is to use simple prioritization classifications, such as the MoSCoW model, where customers rank each backlog item as Must Have, Should Have, Could Have, and Won't Have. The drawback with priority classification is that often customers simply mark everything as "Must have" or "Should have" priorities.

As stated, Agile teams would much rather provide a sizing for a backlog item rather than a detailed hourly estimate of the work required to complete the backlog item. It turns out that combined with some additional techniques, such as velocity calculations, sizing can be even more accurate and much easier to provide by the team. During backlog sizing, teams are asked to use an arbitrary numbering scale. For example, some teams use the Fibonacci sequence (1,2, 3, 5, 8, 13, ...) [minus the first 1 in the scale], some others use the prime number sequence or even simply powers of 2 (1, 2, 4, 8). Some additional teams have very simple scales such as (1, 2, 4, 8, too big). In all cases, however, sizing works in the same way. First a team will agree upon a backlog item that is of "average" size, assigning that item an average number in the scale. Every other item in the backlog will be compared to this average item to see if it is larger or smaller and by what magnitude.

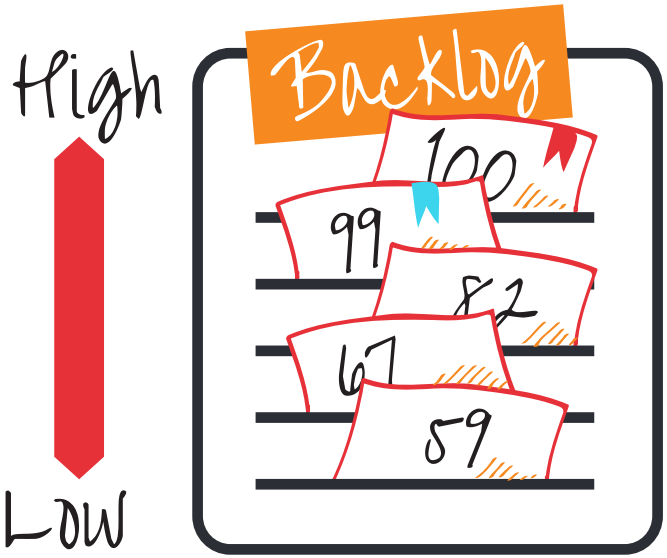


FIGURE 3 - PRIORITIZED BACKLOG

PRIORITIZED AND SIZED BACKLOG

For example, suppose the team chose “Enter Customer Details” as a backlog item that they found to be “average” and assigned the number 5 on the Fibonacci scale. Another backlog item might be “Enable Authentication” and the team considers this backlog entry to be a smaller amount of work than average, in fact about twice as small, and thus assign the number 2 to its size.

FIGURE 4 - ESTIMATE IN SIZES IN RELATION TO DIFFERENT ITEMS

For another backlog entry, “Encrypt Credit Card Number,” the team considers this to be more than twice as big as the “Enter Customer Details” item, and assigns the size of 13.

Sizing works well because teams have a much easier job being accurate regarding the size of a backlog item compared with creating an hourly work estimate. Teams can then use this size information to calculate a team’s velocity, which happens to be an extremely accurate measure of predicting future work. Please refer to the Iteration Velocity and Capacity charter for more information.

How the TeamPulse team does it – Backlog Prioritization

The TeamPulse team uses MoSCoW (Must Have, Should Have, Could Have, Won’t Have) priority classifications to help determine what to do next. Of course, the team’s backlog changes almost daily–new items are constantly entered via our TeamPulse Ideas & Feedback Portal, as well as from internal stakeholders. That’s why we re-prioritize everything at the end of each iteration, resulting in a fluid and constantly changing target (why we absolutely LOVE agile).

When we prioritize we also take into consideration the “cost” (i.e. estimate) of that feature. This is because a feature may seem very important, however, it could be so large that nothing else in an iteration can get done, in which case we reduce its priority. We have found we cannot truly prioritize the backlog until looking at both value AND cost/risk.

Chapter:

UNCERTAINTY AND MATURITY

Not all requirements will be ready when your building commences, so take into account uncertainty and maturity. For example, several items at the top of your prioritized backlog may simply not be ready to implement. These items might be the highest priority, however they may not be completely defined (immature) or may not be valid (uncertain).

To compensate for this, many Agile teams track the certainty and maturity of each backlog item to help them monitor which backlog items are ready to begin development. For example, teams may choose to begin developing items on the backlog that are certain and mature in addition to those at the top of the backlog. Further, team members may allocate time during a planned iteration to validate such items or to more completely define backlog items so work can start on high-priority requirements.

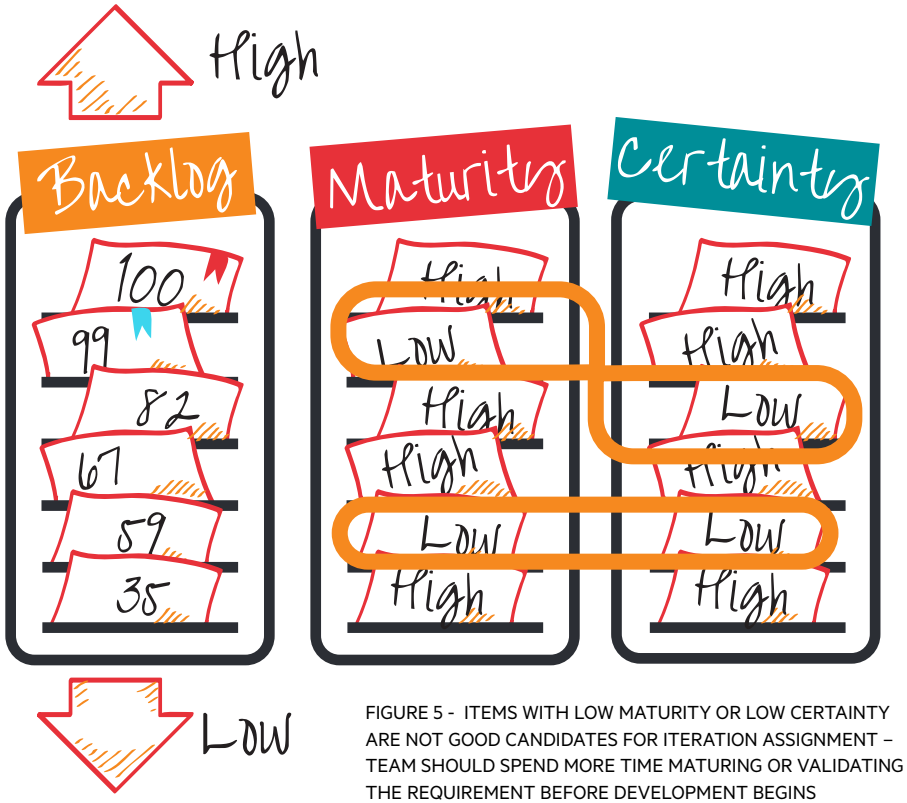


FIGURE 5 - ITEMS WITH LOW MATURITY OR LOW CERTAINTY ARE NOT GOOD CANDIDATES FOR ITERATION ASSIGNMENT – TEAM SHOULD SPEND MORE TIME MATURING OR VALIDATING THE REQUIREMENT BEFORE DEVELOPMENT BEGINS

Chapter:

ITERATION VELOCITY AND CAPACITY

When iteratively planning an Agile project where the team partitions work into timeboxed iterations, understanding capacity is very important. Planning an iteration is very much like using a bucket to scoop water out of a pool, where the pool represents the complete set of backlog items on your project and the bucket represents the amount of work that can be accomplished in a single iteration. Understanding how much water a bucket holds will give you an idea of how many buckets are required to empty the pool. Similarly, understanding how much work your team can deliver in an iteration will give you a good idea of how long the project will take.

An iteration's capacity is the amount and size of the backlog that can be completed in a single iteration. Refer to the following table:

TABLE 1 - EXAMPLE OF ITERATION VELOCITY

Backlog Item	Backlog Item Size	State
Item 1	3	Done
Item 2	5	Not Done
Item 3	1	Done
Item 4	5	Done
Item 5	8	Done
Item 6	2	Not Done
Total Velocity	17	

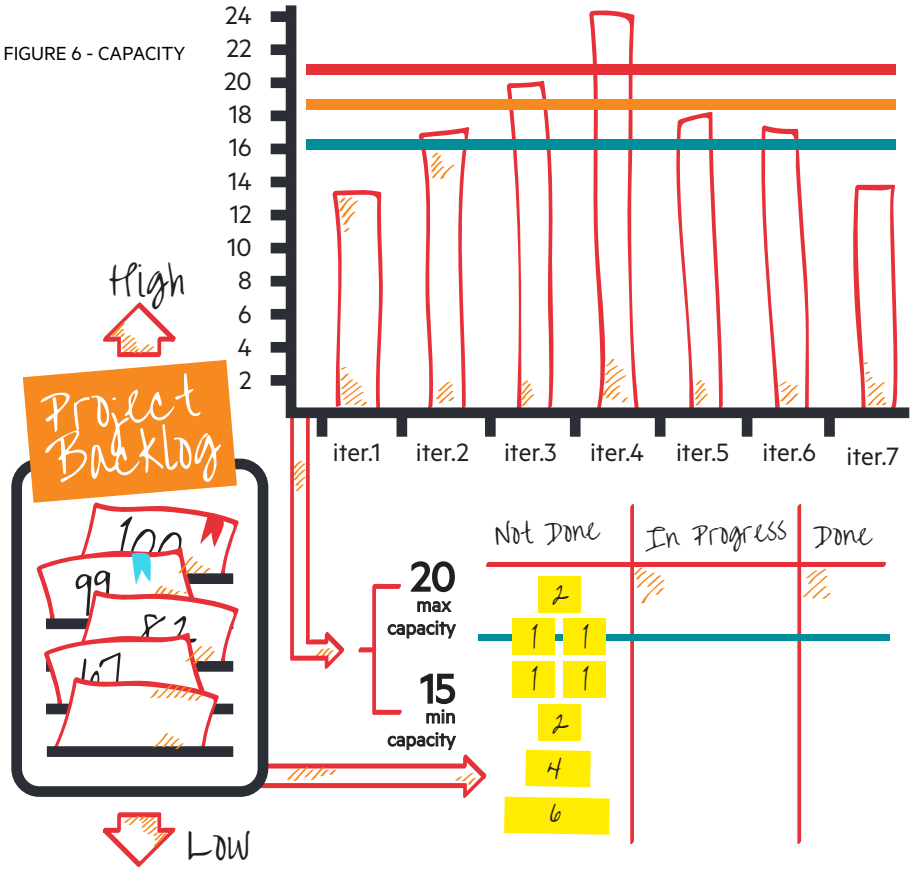
In Table 1 we can see that six backlog items were added to an iteration, two of which were not completed within the iteration. In this case, the team would consider their velocity to be 17 after adding up all of the estimated sizes for all of the backlog items that were complete in the iteration. This number will then act as a guideline for scheduling work for the next iteration as the team has recognized that they scheduled too much work in a single iteration.

Of course, not every iteration will have the same velocity, however, over time the average iteration velocity should begin to become an increasingly

ITERATION VELOCITY AND CAPACITY

accurate predictor of how much work the team can produce in a single iteration. Many teams like to keep a high and low capacity for their iterations, as demonstrated in Figure 6. This allows teams to accurately predict how much work they can accomplish in future iterations and set appropriate expectations with stakeholders.

FIGURE 6 - CAPACITY

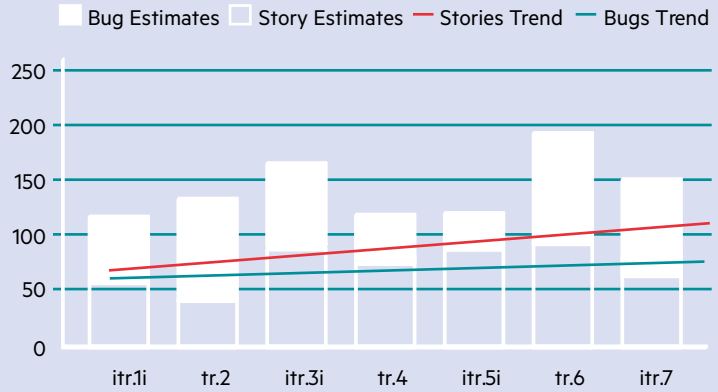


How TeamPulse does it - Capacity Estimation

Our capacity is measured by how many story points the team can deliver in a two-week period. The team uses story points rather than "perfect days" or hours during estimation—as story points are arbitrary and focus on sizing rather than time estimation, which would quickly erode into a "point" system anyway.

We use the built-in TeamPulse reports to help us understand what this capacity is and predict what we can deliver in future iterations.

Here is the exact TeamPulse report we use to help us predict our team capacity:



As you can see, the team got much more efficient over a course of a release – the biggest reason was due to how the team ran its retrospectives which would gradually interject learning and efficiencies into the development process.

Chapter:

DECOMPOSITION INTO TASKS

Your backlog is normally comprised of requirements in the form of User Stories which depict the resulting value required by the users of the system. During iteration planning, teams take the highest value requirements and assign them to the current or next iteration, filling the iteration to capacity. Agile teams typically do not decompose requirements into work tasks until they are assigned to an iteration. Agile teams spend the first few hours of each iteration going through iteration planning, and a part of that process is the decomposition of requirements into work that can be completed by team members. This is more commonly referred to as just-in-time work decomposition.

Task decomposition is not typically done until the requirements are assigned to an iteration. This ensures that the entire team can collectively review and discuss each requirement and determine a common approach to proceed. Task decomposition prior to iteration planning may result in considerable rework and waste.

Just as there is much benefit in the entire team reviewing and understanding each backlog item/requirement, there is similar value in the whole team working to collectively decompose the work required to deliver each requirement. For example, junior members of the team get exposed to the knowledge and techniques of more experienced team members. Teams can also reduce errors and increase quality by contributing to the implementation approach together. They can further help each team member have a strong understanding of all parts of the system regardless of which part they worked on.

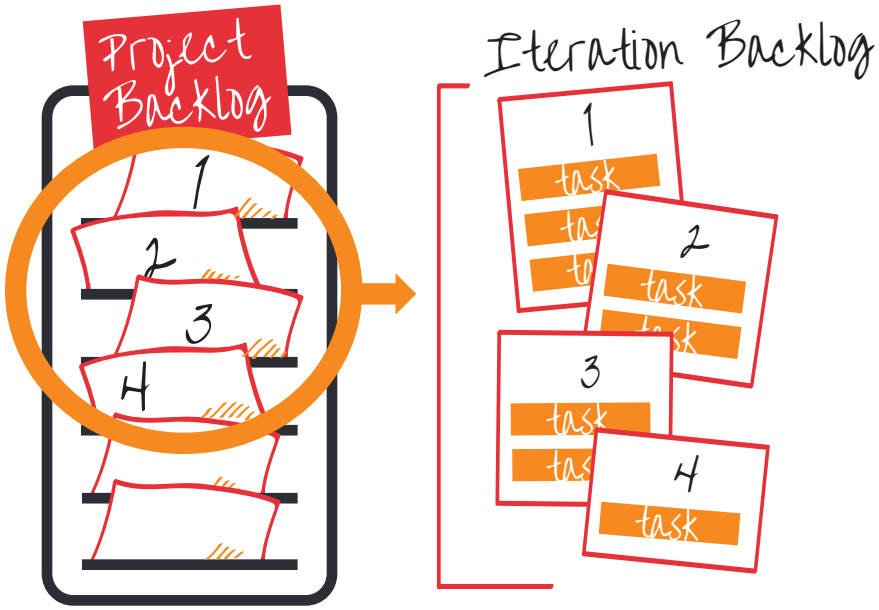
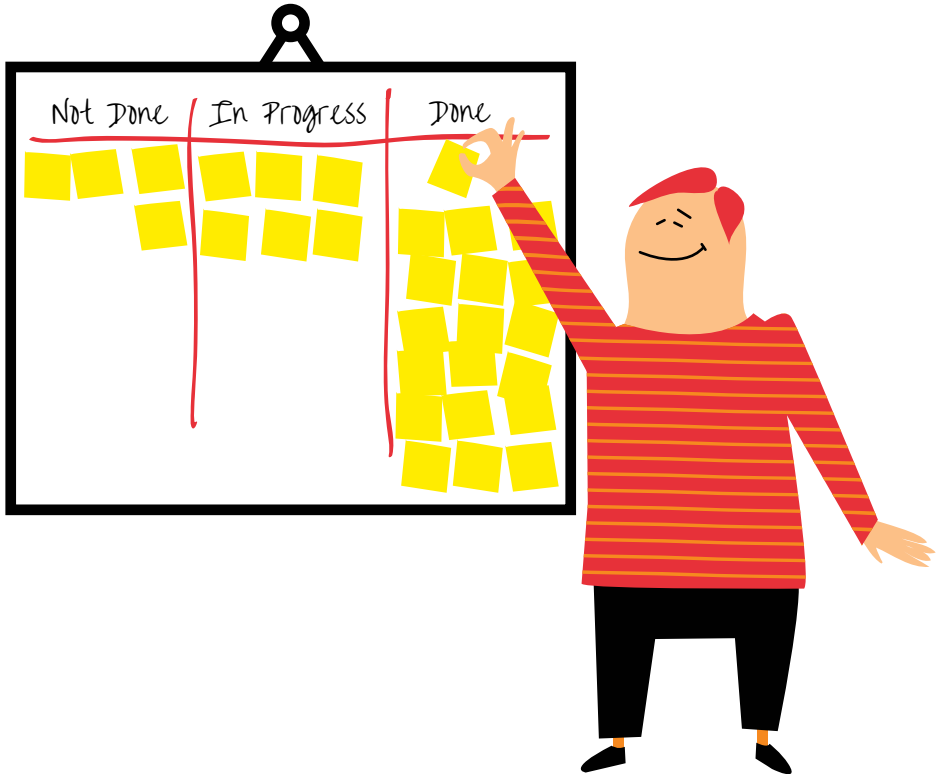


FIGURE 7 - ITERATION PLANNING ASSIGNS ITEMS FROM THE PRODUCT BACKLOG TO THE ITERATION BACKLOG AND DECOMPOSES INTO TASKS

Chapter:

AVOID ASSIGNING WORK



Agile teams not only timebox their work and decompose requirements into tasks "just-in-time," they assign tasks to developers only when justified by their workload. In fact, instead of having a project manager assign work to team members, Agile teams much rather have team members assign work to themselves; a technique called "pulling work" as opposed to "pushing work." Agile teams want to ensure that they are working together as a team as much as possble. As work needs to be done, team members are encouraged to assign it to themselves. This way they can promptly do the work rather than sitting idle waiting to be assigned tasks.

Agile teams follow this practice to keep the amount of work-in-progress as low as possible. Work-in-progress is the amount of work that has been started, but not yet completed by the team. An Agile team values "done" work. Having team members pull work when they need it prioritizes completing existing work first and naturally acts to balance the workload across the team.

Chapter:

RELEASE PLANNING

FIGURE 8 - RELEASES HAVE A SIMILAR STRUCTURE TO AN ITERATION

Many organizations have the concept of a "release," where after a long period of time a version of the software is released to customers. This contrasts the continual release process where features are rolled out to customers as soon as they are complete. Organizations that still require large release cycles also need to plan for those releases, and not just iterations.

Agile teams that do resource planning follow the same planning pattern that they follow at the iteration level. This means that Agile teams maintain a product backlog and then assign items from that backlog to a release in the same way items would get assigned to an iteration. The only major difference is that the scope and size of the items assigned would usually be much more granular and less defined than what would be assigned to an iteration. For example, most Agile teams maintain a hierarchy of requirements. Some teams use the term "story" to define the granularity of work that they assign to an iteration, and the term "epic" to define the granularity of the requirements assigned to a release. Epics are very course grained requirements that may still not be very well defined, however, they are similarly sized with each other.

The Agile team begins each release with release planning, and ends each release with a production software release. During release planning, large scale

RELEASE PLANNING

requirements (such as epics) are assigned to the release and the team will work to decompose the epics into stories – both of which are sized. Similar to iterations, releases are also timeboxed and have a capacity range to help guide teams on how much work can be done within a release.

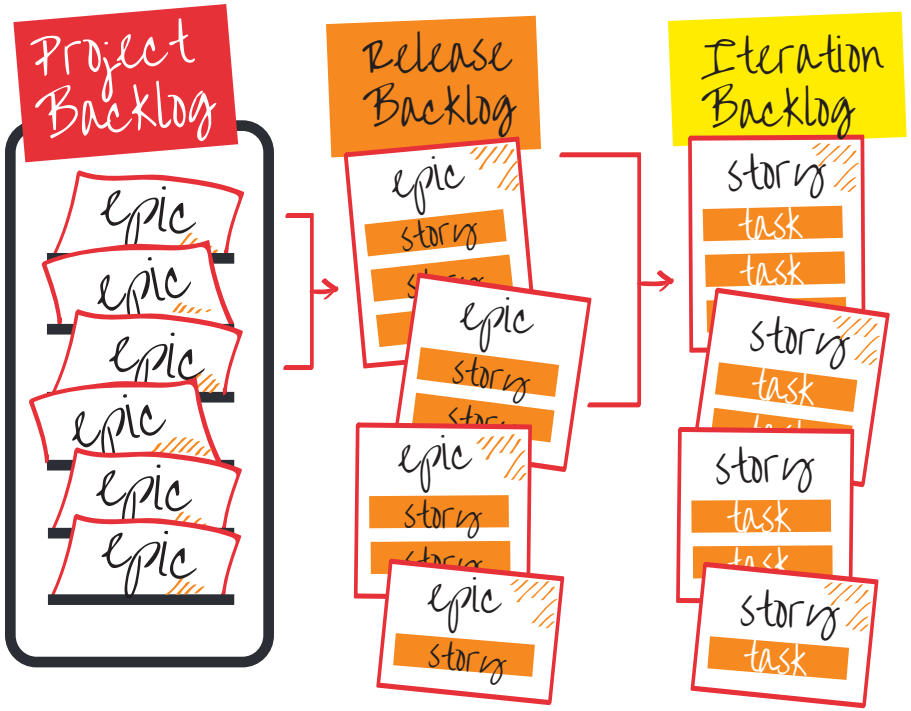


FIGURE 9 - PLAN RELEASES WITH MORE GRANULAR SCOPED BACKLOG ITEMS

Chapter:

COMMON AGILE PLANNING CHALLENGES

Despite trying to achieve simplicity, Agile teams may still run across difficult issues. In this section some of the more common challenges are explored.

Handling Incomplete Work at the End of an Iteration

It is not uncommon for a team to have incomplete work at the end of an iteration. Unfinished work is an important issue to identify as it signals a potential problem with one or more aspects of the team. When an iteration is planned, the team sets an expectation with the customer. When those expectations are not met, the customer could lose faith in the team's ability to deliver, which introduces conditions that make success less achievable. Unfinished work should always be analyzed by the team during every iteration retrospective. This is where the team can better understand why the work was not completed and is likewise an opportunity to chart a better course going forward.

There are only a few things that teams can do to manage unfinished work. First, the team can move the work forward into the next iteration. This is normally done with work that has been started and is close to completion. Work that is not done—and was not even started—is usually moved back to the main backlog and is prioritized and rescheduled with the rest of the backlog items.

Incomplete work normally does not count towards team velocity, however, some teams like to have an earned value approach. Most teams only factor in completed work when calculating velocity since one of the reasons work may be left incomplete is because more was scheduled in the iteration than its capacity allowed.

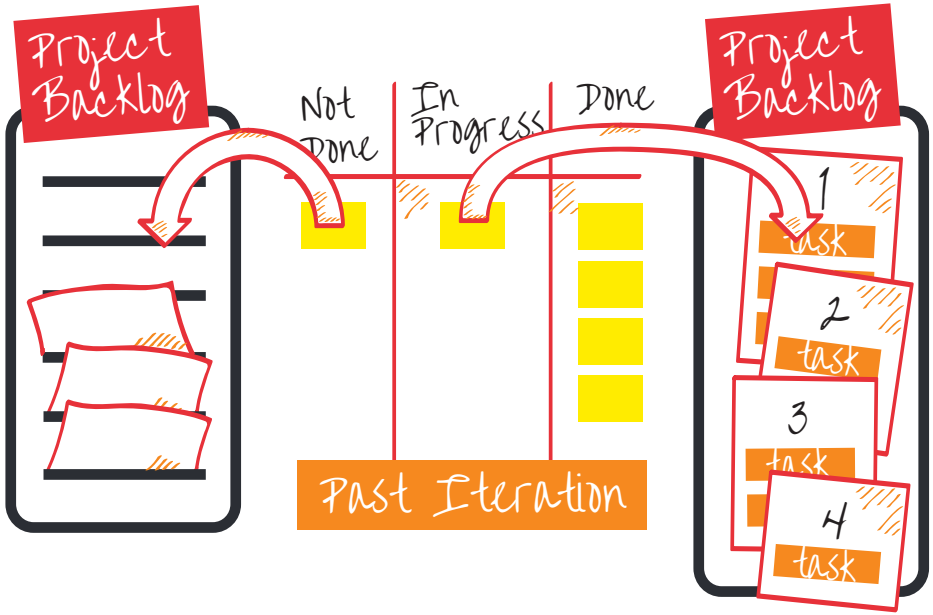


FIGURE 10 - WORK NOT COMPLETED IN AN ITERATION CAN EITHER MOVE TO THE BACKLOG OR TO FUTURE ITERATIONS.

Teams should avoid extending the duration of an iteration to finish incomplete work. To reinforce the hard stop, some teams end iterations mid-week to reduce the temptation of using weekends as iteration buffers. Strict time boxing of iterations is an important tool in helping the team produce more predictable and repeatable results. If work does not fit within an iteration, the team should take the opportunity to assess the reason why so they can eliminate such issues in future iterations.

Chapter:

COMMON AGILE PLANNING CHALLENGES

Handling Bugs

Bugs are a different type of requirement, one the team does not value. Bugs are waste in the eyes of an Agile team. Any time spent fixing a bug is time taken away from producing customer value, and is one of the reasons Agile teams strive for "zero defect" products. Nevertheless, bugs are an inevitable and must be addressed by all Agile teams. But being unpredictable, they cannot be planned as consistently as requirements from an iterative perspective.

Perhaps the most common way to handle bugs on a project is to allocate a particular amount of capacity in the iteration toward fixing bugs. Obviously, iterations early in the project will not need the same bug-fixing allocation as an iteration immediately preceding a production release, so this allocation should be adjusted by the team over time.

It is very difficult for even the most experienced Agile team to forecast bugs and predetermine how they will affect the time allocation in an iteration. Consequently, bugs are pulled into each iteration's bug allocation based on its priority and impact. Since critical bugs can manifest daily, the bug backlog must also be managed daily, a process more commonly known as bug triage. Bug resolution is also very difficult to estimate since teams usually have to work to reproduce bugs and research the root cause before any time estimate for a bug can be made.

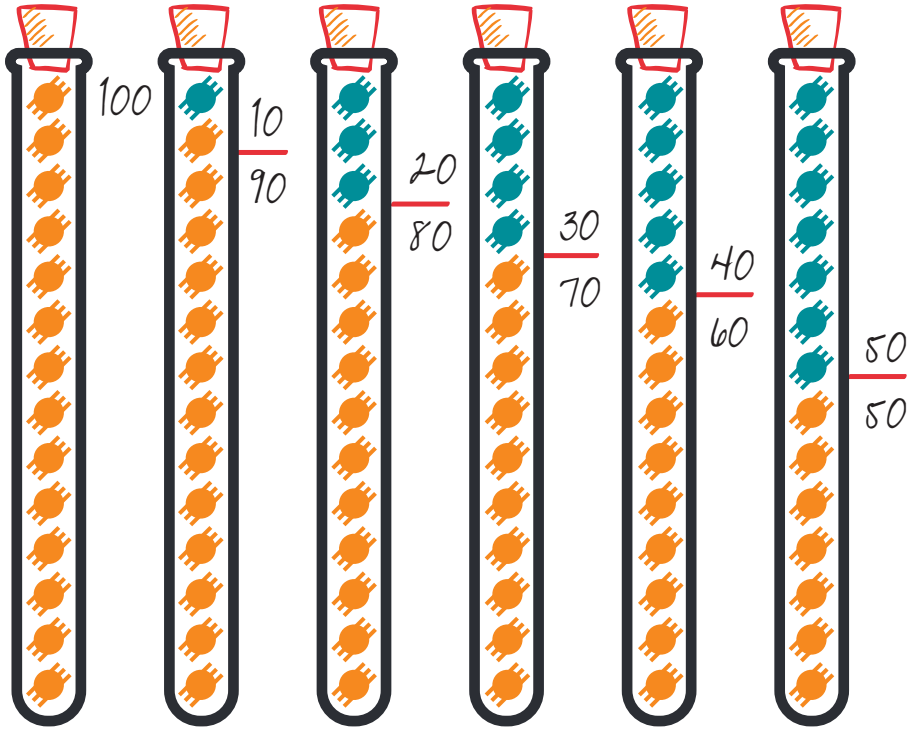


FIGURE 11 - AGILE TEAMS ALLOCATE A % OF THE CAPACITY OF AN ITERATION FOR BUGS

COMMON AGILE PLANNING CHALLENGES

Handing Uncertainty with Spikes

Every project will contain a degree of uncertainty. Agile teams encounter technical uncertainty (what technology, approach or design to employ), as well as uncertain requirements. Uncertainty is usually resolved through a combination of experimentation and further research. Agile teams work to address uncertainty with a "spike." Like almost all things on an Agile project, a spike is a timeboxed amount of time dedicated to addressing an uncertainty. For example, an Agile team may not know the correct approach for a certain technical problem or integration. In this case, the team will schedule a spike time boxed to three or four hours, where one or more members of the team would perform further research or experimentation required to help resolve the uncertainty. Spikes are scheduled in the iteration as any other requirement and decomposed into tasks accordingly.

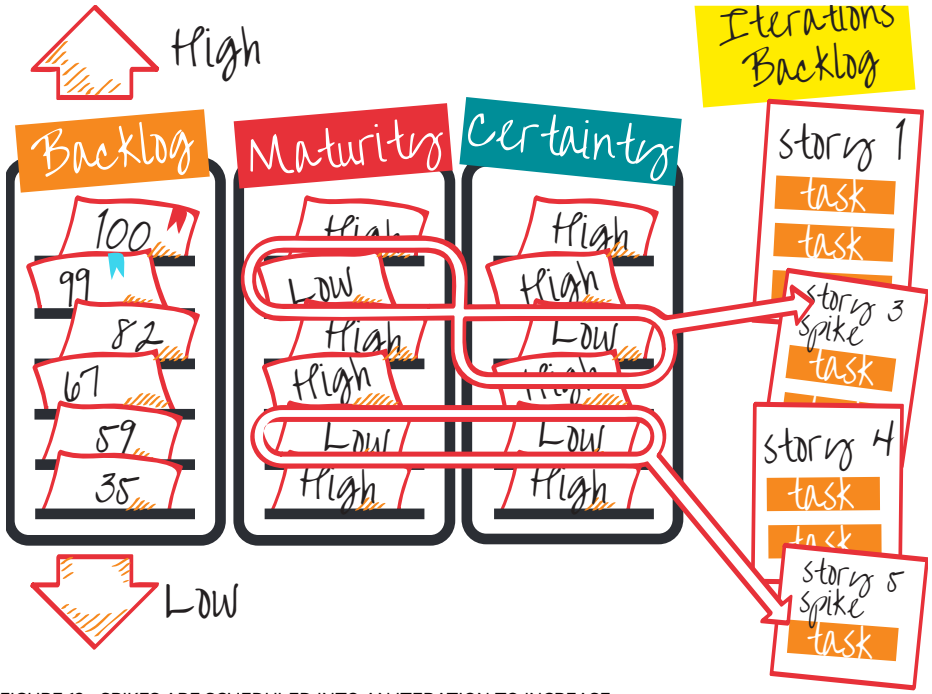


FIGURE 12 - SPIKES ARE SCHEDULED INTO AN ITERATION TO INCREASE CERTAINTY ABOUT TECHNOLOGY OR REQUIREMENTS.

Chapter:

COMMON AGILE PLANNING CHALLENGES

No Time for Agile Meetings

Agile teams focus on removing waste and sharing knowledge. The following four meetings are critical to this process and should never be skipped because they are a critical best practice in Agile planning:

1. Daily standup (Daily Scrum in Scrum)
2. Iteration planning (Sprint planning in Scrum)
3. Iteration review (Sprint review in Scrum)
4. Iteration retrospective (Sprint retrospective in Scrum)

Some teams who adopt Agile struggle with process design which of necessity allocates meeting time to maximize communication and collaboration within an iteration. It can be tempting to skip these meetings to instead complete work, but beware. This can have a detrimental impact on the end result and hamper the team's ability to achieve its project goals.

Many teams choose to stop doing iteration retrospectives, which are meetings held at the end of an iteration where the team gets together and openly share what they believe went well, and specifically, what needs improvement. Retrospective meetings are perhaps the most important part of any Agile process as they are a direct conduit to delivering customer value more effectively. In fact, the retrospective meetings and the process improvement

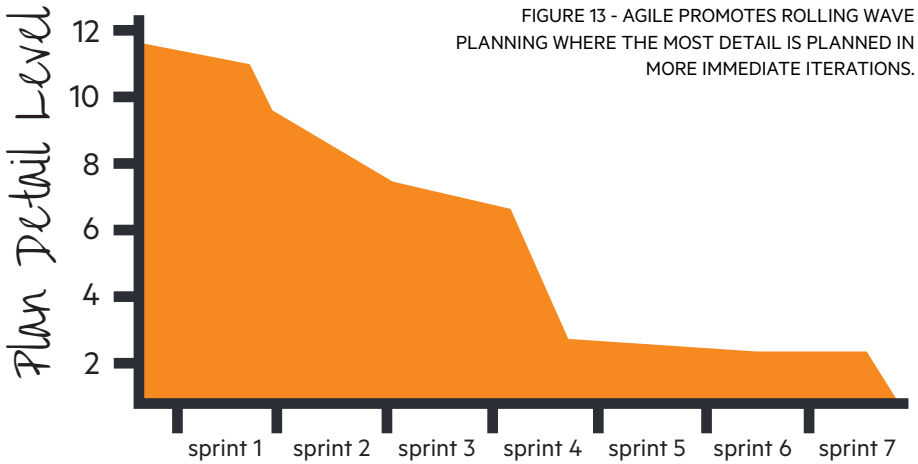


FIGURE 13 - AGILE PROMOTES ROLLING WAVE PLANNING WHERE THE MOST DETAIL IS PLANNED IN MORE IMMEDIATE ITERATIONS.

ideas that result in action are exactly what makes the team "agile." Retrospective meetings are the critical mechanism to help drive continual and ongoing process improvement on a team.

Another potential meeting casualty is the daily standup, known in the Scrum methodology, as the Daily Scrum. The daily standup is typically held at the same time and location every single day and is strictly time-boxed. Each member of the team answers three questions.

- What did you do yesterday?
- What will you do today?
- Are there any problems that are getting in your way?

COMMON AGILE PLANNING CHALLENGES

Daily standups provide a mechanism to ensure that each team member has an understanding of what work has been done and what work remains. Daily standups are not status meetings and they are not meant to inspire discussion or problem solving—they are a mechanism for the team to share state and to uncover problems inhibiting their ability to achieve their goal.

Another version of the daily standup focuses more on the flow of work rather than individual inhibitors. Instead of the team answering the three questions above, the team looks at each of the items that are in progress and quickly discusses what can be done to usher the items through the workflow.

Again, the daily standup is an extremely important tool for teams to help identify issues during the development process providing the team with the opportunity to remove those issues before they impact delivery. Without these meetings teams can easily miss opportunities to resolve issues as quickly as possible.

How the TeamPulse Team Does It - Review Meetings

At the end of each two-week iteration, we host a sprint review to which we invite as many stakeholders as possible, including clients. It is really important for us to directly engage users as their feedback is absolutely critical to our process. During the review meetings, the facilitator, usually the product owner, starts by reviewing what the team intended to do and what it actually accomplished during the sprint.

Next, the team goes on to review each story delivered in the sprint and has a discussion with all invited about what could be changed, what could be added and if the feature provides enough value. All of the changes and feedback are captured as new stories and fed back into the product backlog to be prioritized with the whole.

Having high bandwidth communication with clients is absolutely crucial to every Agile team. That's why in our team we don't limit ourselves only to the bi-weekly review meetings. We use TeamPulse Ideas & Feedback Portal to provide external stakeholders with a 24/7 direct line to our team where they can submit feedback, ideas or bug reports. This way, we make sure that all feedback is captured and centralized in our system, so we don't lose something important.

Chapter:

COMMON AGILE PLANNING CHALLENGES

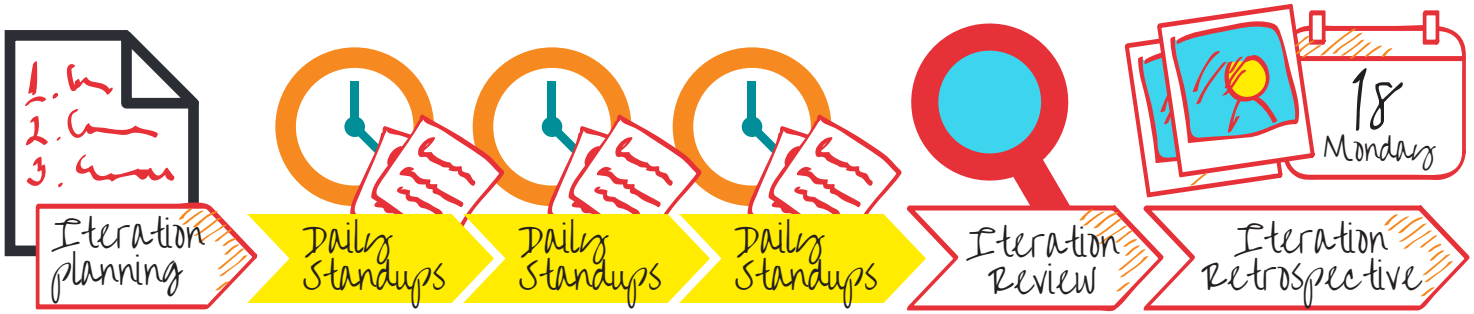


FIGURE 14 - MINIMUM MEETINGS FOR A HEALTHY AGILE TEAM

Another critical meeting is the iteration review meeting. During these meetings the team meets with the product owner to demonstrate what was accomplished during the iteration. This meeting is simply a demonstration rather than a presentation. During this review meeting, the project is assessed against the iteration goal that was determined during the iteration planning meeting. This meeting is a critical feedback loop with the team. Neglecting to perform this meeting means that valuable feedback is ignored. Some teams feel that they haven't produced enough during an iteration to warrant an iteration review.

However, Agile teams strive to get feedback on everything they produce as quickly and as early as possible to help ensure the appropriate value is being targeted and delivered. These are essential if the team is to make beneficial course corrections that may come out of the review meetings.

Perhaps one of the most important meetings of an Agile team is the iteration planning meeting. During the iteration planning meeting the business sponsor/users (referred to as the Product Owner in Scrum) describes the highest priority features to the entire team. The team has the ability to ask questions so that they can appropriately define the work they need to accomplish. In addition, during this meeting the team will help determine the goal for the iteration, helping to drive decision making and focus during the iteration. The iteration planning meeting is critical for a number of reasons. First, it helps ensure that the entire team has a crystal clear understanding of the user requirements. It facilitates direct interaction with the users instead of relying upon documentation alone for the information transfer. Without this forum for communication and alignment, there is a strong chance the team will misunderstand project requirements. This cascades to poor decomposition of work, poor sizing and estimation, and most importantly, the team risks missing the opportunity to provide optimum value possible to its users.

Chapter:

COMMON AGILE PLANNING CHALLENGES

It's hard to change habits and mindset

People are naturally resistant to change and you might find that for most team members it's hard to embrace new concepts and processes. The best advice is practice, practice, practice and … patience. Don't get disheartened if things don't work out from the start. The majority of the teams struggle when they start switching from Waterfall to Agile because there are major areas of mismatch between both processes.

HIRE AN AGILE COACH

To make the transition to Agile smoother many teams hire Agile coaches to guide them through the process. I recommend you do this after you have put some effort in and you have successfully adopted some basic processes. This way the Agile coach will help you build on and solidify what you have learned by yourself and you will be able to see faster gains. Plus you might find out that you don't really need to hire a coach because your team is doing pretty well on their own.

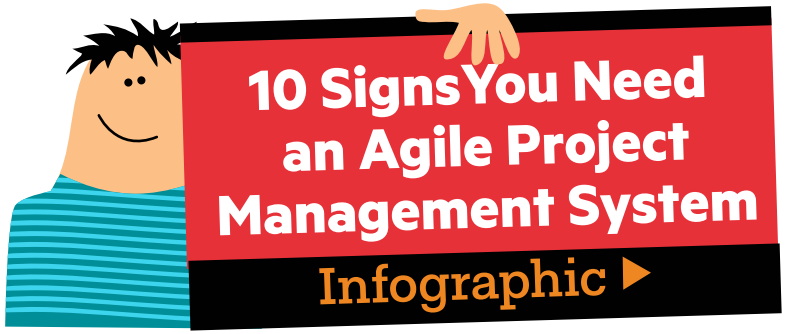
IMPLEMENT A PROJECT MANAGEMENT SYSTEM

Another way to make the adoption of Agile processes faster and smoother is implementing a project management system, like Telerik TeamPulse. The main benefits such tools provide for Agile teams are:

- They are tailored to Agile processes and facilitate the use of the most major Agile artifacts, like User Stories, Backlog, Personas, and Iteration Planning.
- They have built-in guidance system that pinpoints areas where the team is deviating from Agile best practices and gives tips on how to improve.
- They offer A natural way to involve the customers and external stakeholders in the development process through things like online feedback portals.

While most teams can benefit from an Agile project management system, the need greatly varies. Have in mind that each team is unique and you should take into consideration multiple factors before you invest in a tool.

If you are not sure if you need an Agile Project Management system, we recommend you review the following infographic:



CONCLUSION

While there are different valid approaches to development, the Agile approach has legions of fans for quantifiable reasons. It's an efficient method to organize complex development work into executable chunks while likewise empowering all stakeholders to participate in the process. It leverages the power of collaboration to achieve results superior to those achieved by those working in isolated silos. Client participation means that desired functionality is assured, and review guarantees both replicable results and accountability from all quarters. Agile takes discipline and good prioritization skills to stay true to method, but the results speak for themselves.

Having iterations that are too long (many months, for example) require much more planning and have a higher risk of slippage and error. In addition, the feedback loop between the customer and team is much longer providing a greater opportunity for the team experience rework due to errors or omissions.

However, having iterations that are too short can also cause problems. With every iteration comes some overhead—specifically iteration planning—reviews and retrospective meetings. In addition, short iterations leave teams struggling to produce something of value before time expires.

Choosing the right iteration length is very much about finding the right balance between the team and the customer, so be ready for some trial and error. Two-week Iterations are very common among Agile teams, so this is a good place to start for those new to the Agile development methodology.

Utilizing iterations also helps combat two very common and human conditions that we referred to earlier—"The Student Syndrome" and "Parkinson's Law."

ABOUT THE AUTHOR

Follow him on Twitter @JoelSemeniuk

Joel Semeniuk is a founder of Imaginet Resources Corp., a Canadian based Microsoft Gold Partner. He is also a Microsoft Regional Director and MVP Microsoft ALM and has a degree in Computer Science. With over 18 years of experience, Joel specializes in helping organizations around the world realize their potential through maturing their software development and information technology practices. Joel is passionate about Application Lifecycle Management tooling, techniques and mindsets and regularly speaks at conferences around the world on a wide range of ALM topics. Joel is also the co-author of "Managing Projects with Microsoft Visual Studio Team System" published by Microsoft Press, as well as dozens of other articles for popular trade magazines on agile adoption, involvement and collaboration between roles.