Chapter 5

Tracking Agile Project and Reports

Reporting can be a great way to update people about a project's progress and to prepare for important decisions in the future. One type of reporting that businesses often take part in is agile reporting, which uses software and specific methods of analyzing and displaying data. If you work in project management or in a workplace that completes a high volume of projects, you might consider learning about agile reporting so you can use it to benefit your team. In this article, we define what agile reporting is and explore some steps for how to complete it.

What is Agile reporting?

Agile reporting is a technique for analyzing and sharing information about a company using specific software and methods of reporting. Many companies use agile reporting to review their performance because it offers several ways to track progress and share updates with stakeholders.

There are **two main categories of agile reports,**

the **first of which considers how to share the details and progress of a particular project with people who don't take part in the** production process.

The **second kind of Agile report focuses on more specific details and provides analysis that team members can use to inform their future decisions, such as creating budgets and setting deadlines.**

Why is agile reporting important?

Agile reporting can be important for many reasons, such as **the ability it gives project managers to communicate with people who aren't involved in a project about their progress.** This can be especially valuable to companies that complete projects for stakeholders frequently, as they can use an agile report to inform everyone who might be interested in a project about how it's going. **Another reason why agile reporting can be important is that it can ensure that everyone on a team has the same information regarding a project.**

For example, a team might create an agile report to determine how much work they typically complete in a certain period of time and use that information to set new deadlines according to their trends in progress and share them with all members.

Step-By-Step Guide to Agile Planning- Drafting And Executing Projects:

With the ever-growing competition in the market and a need for perfection, companies look for ways to improve their processes for better results. One such way of improving performance in Agile planning.

Agile panning is used by a whopping 71% of organizations to manage their projects. It has been proven that agile projects have a faster delivery rate when compared with other methodologies of project planning and management.

What Is Agile Planning?

The agile project plan is an innovative way of breaking down your project into smaller chunks for quick and efficient delivery. The project planning methodology relies on iterations or sprints to estimate work.

Agile planning uses an **iterative approach** where sprints are repeated to increase the productivity of the team. Iterations or sprints are work units with focused aims and objectives and a definite time limit. A sprint is usually 1-3 weeks long wherein the team members focus to accomplish a planned set of tasks.

With agile planning, teams can identify the tasks to be achieved in each sprint. The iterative process makes it easy for teams to stay on track and complete their tasks in time.

Agile planning identifies the **efficiency of resources** and **creates optimum roadmaps** for the efficient completion of projects.

Let's take a look at an example to understand the basics of the concept more clearly.

Let's say you have to write an article of 1000 words, just like this one. You have 2 days to complete the whole article. Now, the traditional modes of project planning suggest that you complete the article in 2 days and publish it only after it is complete.

Agile project plan, however, suggests that you write 500 words on day 1 and publish those first to attract readership. You can complete and publish the rest of the article on the second day.

Guide To Agile Planning

Agile planning takes place in five levels. We will discuss the levels and explain the details in the forthcoming paragraphs.

Levels of Agile Planning

An agile development plan has several levels. These levels are:

1. Product vision
2. Roadmap
3. Release plan
4. Iteration plan
5. Daily stand up

Product Vision:

Product vision forms the first level of agile planning which is the long-term goals for a product. Teams work together to make sure that their plans align with the end goals. This helps team members stay focused to achieve a shared objective.

This level is representative of the 'big picture' that a product aims to achieve. the planning at this level is more generic and focuses on product information rather than the details of the product.

2. Roadmap

The second level of agile planning is planning the roadmap. The level focuses on strategizing ways to achieve the vision laid out in the first level. In this step agile team members identify a roadmap that lists all the details of the plan.

Roadmap highlights milestones to be achieved and defines the steps following which the organization will move closer to its goal. Team members when and how the project will move towards completion.

3. Release plan

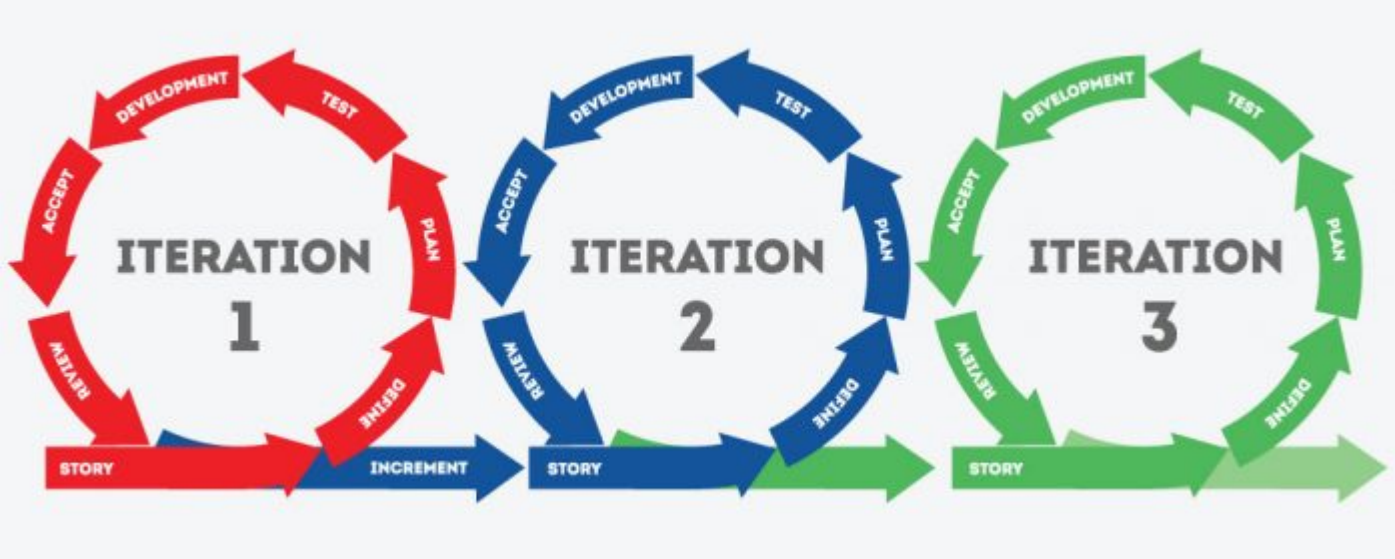
A release plan is the third level of the agile development plan. Agile release planning identifies how much of the project will be delivered to the customer on the specified deadline.

At this level, the personnel of the company works together to plan the release of deliverables. The members involved in this phase include scrum

master (the one who facilitates the meeting), the owner of the product (who shares the product backlog), agile team members (provide valuable insights into technical dependencies and feasibility), and lastly, stakeholders.

The goal in agile release planning is to produce a backlog of the work and plan future releases.

4. Iteration plan



What is iteration planning in agile? Iteration planning is the 4th level of agile planning where all the teams involved in the process start to come together and plan iterations or sprints.

Agile teams plan sprints via agile sprint planning. An iteration or sprint is a specific time period during which changes are implemented in a project. The team members plan for the upcoming 1-4 weeks.

Companies hold a meeting for agile sprint planning. In this meeting, the team members determine the volume of backlogs they can produce during the next iteration.

What is vision planning in agile? Vision planning in agile allows a team to visualize the process and identify the steps for better results in the future.

The team creates development iterations keeping in mind the product roadmap. To make sure that the project flows smoothly, companies organize sessions to evaluate performance at the end and start of each iteration.

To understand what is iteration planning in agile in detail, let us take a look at the step-by-step process of iterative planning.

- **Step 1:** Identify and prioritize features from the product backlog
- **Step 2:** The development identifies features that can be accomplished within the next iteration.
- **Step 3:** The team identifies the tasks required to complete for the next iteration. Time and budget are also estimated in this step.
- **Step 4:** Features are added or removed from the product backlog.

To make sure that your sprints are planned to perfection and there are no hiccups, you have to consider the capacity and velocity of your team members.

i. Capacity: Capacity refers to the capability of your team to complete a sprint. You evaluate the performance of each team member and determine how much he/she can achieve. you have to take into account the possible hindrances that might affect the working of your team members such as illness, training, workshops, etc.

ii. Velocity: Generally, velocity is the ratio of displacement over time. It determines the rate at which an object moves towards its destination. Similarly, velocity in project planning is a measure of how much your team can achieve in a given time limit.

5. Daily Standup

The daily standup is the last level of Agile planning. At this level **team members gather together and discuss the schedule for the next day, their achievements for the day before, and evaluate problems if any.**

Teams get together and communicate their role in the accomplishment of the iterations. It is a face-to-face session of 15-30 minutes where team members discuss their progress.

All team members can participate and voice their concerns for the next iteration, explain their input in the process, or share the things they have to do.

Scaling Agile Planning

According to a research titled 'Scaling Agile Processes: Five Levels of Planning' by Hubert Smits, the typical time limit for each level is as follows:

- Product vision – Product owners plan a vision for the product **once a year** or so

- Roadmap – Planned by product owners **twice a year**
- Release plan – Product owners and teams plan the releases for **four quarters of a year**
- Iteration plan – Planned by teams **twice a week**
- Daily stand up – Conducted by individual team members **daily**

Facilitate Retrospective, Making Team Decisions and Closing out Retrospective:

How to get Optimal Value of a Retrospective – Part I

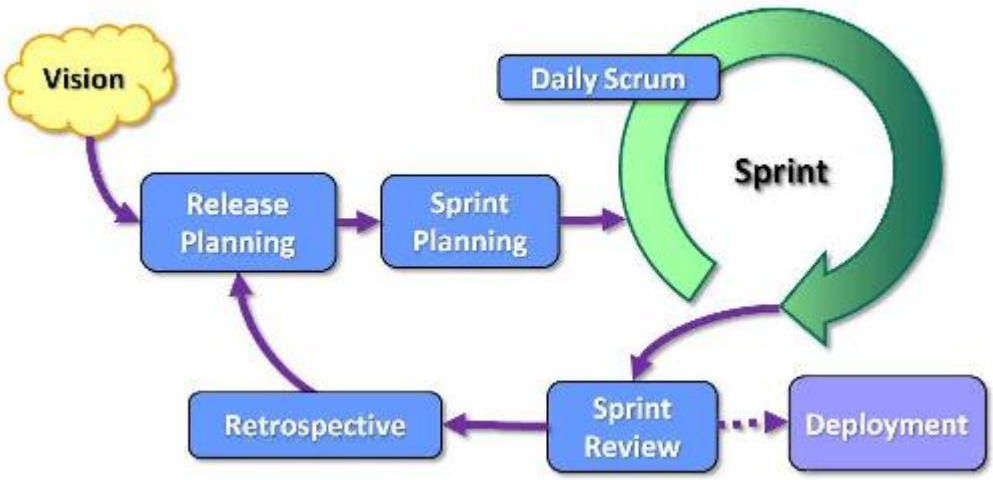
The idea of the retrospective, is that it is a window for teams to inspect and adapt, to learn about what works and what does not work, and to find better ways of working together and with their product owner, ever striving towards the lean principle of kaizen (continuous improvement). For this to occur the team must keep these primary goals in mind:

1.Open and Honest Communication

2.Group Understanding

3.Executable Action Items

The last of which is the defined deliverable for the retrospective, the Executable Action Items. Without this you are simply clearing the air, either griping or congratulating without point or purpose. This artifact is the ultimate output of the retrospective meeting, a ratified plan on what elements of a teams working environment or practices to change in order to **improve performance over past iterations.**



The Executable Action Items are a very short check list or to do list of suggested ideas, also known as experiments, for improving the performance of the team. This list is derived from a subset of all the issues discovered in the retrospective meeting. The list of open issues are scoped down to a short list of high priority items, less than 3, that the team wants to address in the next iteration.

I call these Executable Action Items because each item should require some action by the team, should be executable, measurable and demonstrable. Sometimes this is as simple as modifying the existing team agreements. Sometimes this requires a change in practices. Sometimes this requires new patterns of collaboration with external sources such as the Product Owner, or external teams like IT, or Release Management.

Every retrospective should produce one or more clear Executable Action Items that are sourced from the team, agreed upon by the team, and most of all committed to execution by the team. This artifact should be recorded for historical reference on a wiki or other shared resource, as well as on a Big Visible Chart in the team room or work area.

This BVC should be referenced every day during team standups to verify the team is consistently working towards improving their processes and working habits.

So, to answer the first part of our readers question, the answer is to make sure you generate a short list of Executable Action Items as your retrospective arti fact, post it in the team area for all to see and record it on a team wiki for historical reference.

Successful Retrospective Facilitation – Part II

The second part of the question posed by our reader is more difficult to explain briefly as it involves how to plan and facilitate the retrospective meeting.

There are literally thousands of blog posts and publications on Agile Retrospectives. When I did a search on google I found 165,000 hits; a staggering number to sift through. Fortunately there are a few clear winners in this field. As mentioned above, the primary resource for learning about effective retrospectives is the book by Esther Derby and Diana Larsen, with additional information on each of their blogs. These authors are known in project management circles as the "Retrospective Queens". For every manager, scrum master or otherwise servant leader of Agile Delivery Teams, this book is a must read. Although this is not the first book written on Retrospectives, nor is it the last, it has become a standard that many refer to.

Today my favorite resource for all things on Retrospectives is the Agile Retrospective Wiki. This site is full of collections of ideas from the best in the industry. Many of the entries on this site also appear in Esther & Diana's book. This site is a growing collection of patterns, tools and ideas for giving powerful, fun and successful retrospectives.

Esther & Diana's book outlines a 5 step plan for organizing and facilitiating successful retrospectives. Most of the industry is in agreement and has adopted this strategy. The 5 steps are:

1. Set the Stage

2. Gather Data

3. Generate Insights

4. Decide What to Do

5. Close the Retrospective

Some facilitators try to take a shorthand approach to their retrospectives, skipping step 1: Set the Stage, heading straight into a round robin style inquiry of the team to gather data. While this might work with very small teams that have been working well together for a very long time, it has a number of drawbacks.

It is important to remember that the 5 step plan given above was honed after many years of both failed and successful retrospectives, and post mortem meetings, by professionals throughout the industry. From their collective wisdom we learn that the most successful retrospectives all follow a similar pattern. Unless you are a seasoned facilitator and have lead many successful retrospectives then it is not recommended to stray from this guideline. On the other hand if you find something that works well for your team then I encourage you to share that pattern with others, post a note to Esther or Diana, or on the Agile Retrospective Wiki.

Step 1: Set the Stage

The first step, Set the Stage helps to focus the team on the task at hand, group learning, open and honest communication. As a developer, having participated in many retrospectives, I remember attending more than one as a "prisoner". My mind was focused on the bug or technical challenge that prevented us from delivering a story for the sprint. I did not want to sit and talk about feelings, I wanted to get something done.

A good facilitator will always have a retrospective plan in mind and come to the meeting prepared. Setting the stage helps to break us out of our mental thought patterns and re-focus our energies on the purpose of having a retrospective, 'kaizen' . Don't skip step one, it is there to help you get the most out of your retrospectives.

To set the stage the facilitator wants to thank all the **participants for showing up, having an open mind and willingness to work together to help improve the teams overall performance**. They will need to introduce their plan for the retrospective meeting giving the agenda and a rough outline of the timeline. Next they should elicit some form of verbal response from every team member, whether it is simply stating their name, or a word game where each member defines their current state, or their perception of the sprint events in a word or two. Getting this verbal exercise out of each and every team member is a way of changing their state and informing their body that actual physical participation will be required.

The facilitator should then review the teams working agreements to remind the team and to establish a safety zone for the retrospective meeting. Remember, little value will come from a hostile environment. Every member of the team should be safe and empowered to speak honestly about every aspect of working with the team, the scrum master and the product owner for the past sprint. No judgement shall occur for anything said within the meeting: Everything discussed within the retrospective boundaries is with the spirit of open and honest communication for enhanced collaboration.

Finally the facilitator will want to lead the participants in a trip down memory lane eliciting brief synopsis of the events over the past sprint from the team's memory. This frames the meeting scope and reminds all participants of what happened during their sprint, good, bad and ugly.

Opening the meeting and setting the stage in this way helps the participants to refocus on the task at hand, remember what happened and what was important, and more importantly be present and available for participation in the group exercises.

The resources listed below define many patterns and games you can use for each step, mix it up, try them all, have fun.

Step 2: Gather Data

The next step in the 5 step plan is to gather data. **This is a tricky step to have successful participation by all members of the team.** Over time many different patterns have emerged from leaders in the industry on how to best encourage full participation.

Again, you should try them all; as a retrospective facilitator you need to work hard to keep the process fresh and interesting thereby generating the greatest participation and quality results. In each of these patterns the focus is looking for both positive events (things the team wants to continue) and negative events (things that the team feels they need to improve or manage in order to perform better.)

Previously we mentioned the "**Round Robin"** pattern of inquiry for generating data. What might the drawbacks of this pattern be?

Let us consider a relatively new team who has been working together for only a few sprints. In round robin you gather them around a table and ask each in turn for input, what went wrong, what went well. One at a time you put them on the spot. If you were on this team how would this make you feel?

People have different behaviors, some are very vocal, others less so, some are brash and speak their minds, others are concerned about offending someone… This round robin style has the least chance of success with a new team. With a well seasoned team that has built considerable trust and confidence the round robin style may work, but even then, I have not seen it be effective; not to mention that this serial method of data collection takes too much time to get all the data.

Furthermore, during this round robin style, while one person is giving their answers the other members are not truly listening. They are thinking about what they are going to say when

they are under the heat of the spotlight. Also, consider that while that person is speaking about how terrible the network problems were the other team members may drop that item off their list as it has already been voiced. This action causes you to lose a valuable piece of data — consensus.

For these reasons most of the successful patterns for data collection involve some form of silent brainstorming. Regardless of the patterns used the retrospective needs a strong facilitator to help control the dominant personalities.

In silent brainstorming each team member is given a stack of 3×5 sticky notes and a medium felt tip pen like a sharpie (this forces the notes to be written large enough to be read from a distance while limiting the details on each note.) The team is given a timebox for generating ideas, sometimes a goal is placed before them like "generate 10 sticky notes each". Each team member writes down their ideas that answer the questions posted on the board during the "Set the stage" step. Each retrospective pattern has slight variations on the questions posed.

As the team members work independently to generate ideas there will be a flow, maybe slow to start then gaining more momentum as their brains focus more deeply on the past sprint activities. If you have set the stage appropriately their minds will already be focused on the past sprint and will have been reminded of major events that occurred.

Eventually the flow will slow to a trickle or even stop. Even though the facilitator is time-boxing this activity, if the flow is strong they should hold off on closing the time box, and if the flow trickles to a halt simply ask if the team wants more time or if the time-box should be closed early.

(Slow team members and fast team members)

Now what are the psychological forces at play here? First of all the silent brainstorming method makes all members equal, both the loudly dominant and passive members are given equal voice. The quiet members are given courage to voice their opinions and ideas, and the loud members are forced to constrain their ideas to what fits on the sticky notes. Frequently team members will post their notes on the board as they generate them, sometimes the facilitator will canvas the group and offer to post any completed notes on the board while the team continues to generate data points. Team members who begin to slow down can read the notes already posted and be prompted for new ideas. Usually they will see duplications which emboldens their spirit with the knowledge that other teammates are thinking similarly.

Step 3: Generate Insights

When the time box is closed the facilitator will ask the group to join him at the board to help organize the sticky notes into groups to uncover the themes represented by the generated data, like Network Troubles, Story Content or Acceptance Criteria, Interruptions, Product Owner availability, etc. Usually there will be some amount of duplication, this is consensus, a valuable data point that helps build team cohesion.

When more than one person produces a note with an identical or closely related topic this demonstrates how prevalent or important that topic is. When doing the round robin style the

only way to collect this data is to acknowledge the nodding heads around the table when one person is first broaching the topic. I think that when 2 or more people are affected strongly enough by the idea to synthesize and record it on a sticky note it shows a much higher degree of consensus.

Once all the notes are affinity grouped the facilitator can read through each note or group of notes, leaving an opening for each author to add details thru discussion, or possibly requesting volunteers to explicate complex topics. This is a tricky situation here where the facilitator needs to be careful not to attack or put anyone on the spot. They need to help each team member feel supported, protected and safe so that they can share their ideas without retribution. The ground rule of any brainstorming session is that no judgement is applied when ideas are expressed. The facilitor should make this clear in step 1 Set the Stage.

Once all the idea groups have been discussed the **team will proceed to prioritizing the themed groups by vote and discover the top 2 or 3 items where improvement is desired.** I like to use multi-voting for this stage. In multi-voting, otherwise known as dot voting, each participant is given 3 votes (sometimes more if the number of affinity groups is large). They may place those votes any where they choose. Each vote signifies a topic that is important to someone. After the voting is complete the facilitor tallies the votes and identifies the top 1 or 2 topics for discussion.

Step 4: Decide what to do.

The top topics are given themes (subject names) for easy reference. From these themes the facilitor leads a discussion with the team on how to improve these topic areas. This is once again a form of data collection and it is up to the facilitor on how best to do this, but generally once the dirty laundry has been aired and agreed upon the team begins to feel emboldened. Soliciting ideas on how to improve the agreed upon subjects become less hazardous. Some patterns have the team break into small groups to generate ideas on how to improve a specific topic. Some teams simply hold an open forum. Choose what works best considering the emotional state of the team, the current sprint and the nature of the topics.

Eventually each topic will have 1 or more Executable Action Items attached that address the described issue. The team discusses each of these items and makes a vote on their ability and desire to commit to execution of each action item. Sometimes a person is assigned to an item with the responsibility to make sure that item is completed, occasionally an expected due date is assigned as well. Other times it is simply the attempt to adopt a new practice across the team.

Step 5: Close the Retrospective

Finally the facilitator will close the retrospective. Closing the retrospective helps the team **leave the meeting with a positive feeling that something good will come from all their work.** This helps to encourage them to participate more in future retrospectives.

The final aspect of the retrospective is to generate a BVC(Big Visible Chart) of the Executable Action Items and post it in the team area for reference and discussion during team standups, and throughout the day as well.

Conclusion:

For a successful retrospective the facilitator must be organized and come to the meeting with a plan. They must be strong enough to control the louder participants while generating an atmosphere of trust and safety so that all team members may participate equally. The results of your efforts should create the following:

1. Safety zone for open and honest communication

2. Group understanding thru discovery

3. Team committed Executable Action Items

The final artifact of the meeting is the Executable Action Items, which should be recorded in a team wiki and posted on a BVC in the team work area. There are vast resources available to help you plan and keep these sessions interesting, fun and productive. And remember, always have fun.

5.4 Agile Reports

What Is Agile Reporting? (Definition and How To Complete)

Reporting can be a great way to update people about a project's progress and to prepare for important decisions in the future. **One type of reporting** that businesses often take part in is **agile reporting**, which uses software and specific methods of analyzing and displaying data. If you work in project management or in a workplace that completes a high volume of projects, you might consider learning about agile reporting so you can use it to benefit your team. In this article, we define what agile reporting is and explore some steps for how to complete it.

What is Agile reporting?

Agile reporting is a technique for analyzing and sharing information about a company using specific software and methods of reporting. Many companies use agile **reporting to review their performance because it offers several ways to track progress and share updates with stakeholders.**

There are **two main categories of agile reports**, the first of which considers how to share the details and progress of a particular project with **people who don't take part in the production process.** The second kind of Agile report focuses on more specific details and provides analysis that **team members can use to inform their future decisions,** such as creating budgets and setting deadlines.

Why is agile reporting important?

Agile reporting can be important for many reasons, such as the ability it gives project managers to communicate with people who aren't involved in a project about their progress.

This can be especially valuable to companies that complete projects for stakeholders frequently, as they can use an agile report to inform everyone who might be interested in a project about how it's going. Another reason why agile reporting can be important is that it can ensure that everyone on a team has the same information regarding a project.

For example, a team might create an agile report to determine how much work they typically complete in a certain period of time and use that information to set new deadlines according to their trends in progress and share them with all members.

How to complete Agile reporting

Here are some steps you can use to take part in agile reporting at your workplace:

1. Consider your purpose and goals for reporting

The first step in completing an agile report is to determine **why you want to create one**. This is because a report can often be most effective when it has a specific purpose or goal, such as identifying milestones to track progress or figuring out how quickly a team might be able to complete a certain project. Knowing your reason for reporting can also help you to choose a type of report to use, as certain formats for agile reporting can be most effective when they consider particular types of data.

When thinking about your purpose and goals, you might review details like what aspects of a team you want to improve and why you want to share certain information about a project.

2. Determine what data you want to report

After you consider your reasons for reporting**, it can be helpful to think about the specific data you want to include in your agile report.** As there can be several sources and types of data in a company that leaders might view as important, identifying a particular area to focus on can keep your report specific and clear. For example, if a company wants to evaluate its manufacturing process, it might create an agile report that considers metrics like the time it takes to produce a certain number of units of products. Here are some other metrics or types of data that you might use for an agile report:

- Realism of milestones in a project
- Efficiency of product development
- Quality of items produced
- Presence of defects
- Amount of work left in a project

3. Identify the audience for your report achievement

Considering the intended audience of your agile report can be an especially important step because companies typically use these reports in different ways depending on who they intend to read them. For example, a production team might complete an agile report with the intention of showing it to company leaders to verify their efficiency. A company can also use an agile report to share progress with stakeholders in a particular project.

Thinking about who your intended audience is can help you focus on formatting your report according to who might read it, such as **using technical language** in a report meant for company managers or including **only basic details in a report meant for external readers.**

4. Create milestones for reporting

Another key element of creating an agile report is determining when you want to share the report with your intended readers. For example, **one team might prepare an agile report that they plan to send to stakeholders at the completion of a project, while another team might write a report to track progress that they want to send at a project's halfway point.**

Determining when you want to report can also help you determine which method of reporting you want to use. This is because different types of agile reports can be most effective depending on the stage in a project when they're created.

5. Choose a method

Once you're aware of the details and timeline of your report, you can pick a method for reporting. There are a few different types of agile reports, all of which can provide great insight into a company or team's performance. Here are few common methods you can use to complete an agile report:

- Product backlog
- Sprint velocity
- Item process timeline
- Burndown chart
- Burnup chart
- User profile
- Cumulative flow diagram
- Status timeline

6. Complete and share the report

After determining which type of agile report you want to create, you can input your data to complete the report. This typically involves using a particular software to add data and produce charts or graphs that represent it. Then, you can share your report with whoever you want to read it.

WHAT IS A BURNDOWN CHART?

A burndown chart **shows the amount of work that has been completed in an epic or sprint,** and **the total work remaining**. Burndown charts are used to predict your team's likelihood of completing their work in the time available. They're also great for keeping the team aware of any scope creep that occurs.

Burndown charts are useful because **they provide insight into how the team works.** For example:

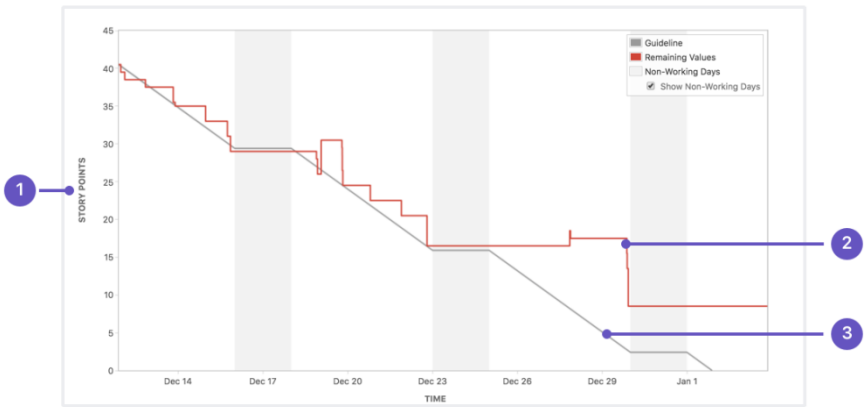
- If you notice that the team consistently finishes work early, this might be a sign that they aren't committing to enough work during sprint planning.
- If they consistently miss their forecast, this might be a sign that they've committed to too much work.
- If the burndown chart shows a sharp drop during the sprint, this might be a sign that work has not been estimated accurately, or broken down properly.

Track team progress with burn down charts

Sprint burn down chart

This report shows the amount of work to be done in a sprint. It can be used to track the total work remaining in the sprint, and to project the likelihood of achieving the sprint goal. By tracking the remaining work throughout the sprint, a team can manage its progress, and respond to trends accordingly. For example, if the burndown chart shows that the team may not reach the sprint goal, then they can take the necessary actions to stay on track.

Understanding the sprint burndown chart



1. **Estimation statistic**: The vertical axis represents the estimation statistic that you've selected.
2. **Remaining values**: The red line represents the total amount of work left in the sprint, according to your team's estimates.
3. **Guideline**: The grey line shows an approximation of where your team should be, assuming linear progress. If the red line is below this line, congratulations - your team's on track to completing all their work by the end of the sprint. This isn't foolproof though; it's just another piece of information to use while monitoring team progress.

Epic burndown chart

This report shows you how your team is progressing against the work for an epic. It's optimized for Scrum teams who work in sprints and makes tracking easier. Here are some of the ways that you could use an epic burndown chart:

- See how quickly your team is working through the epic.

- See how work added/removed during the sprint has affected your team's overall progress.
- Predict how many sprints it will take to complete the work for an epic, based on past sprints and changes during the sprints.

Benefits of agile project management

1. High product quality

In Agile development, testing is integrated during the cycle, which means that there are regular checkups to see that the product is working during the development. This enables the product owner to make changes if needed and the team is aware if there are any issues.

- Defining and elaborating requirements just in time so that the knowledge of the product features is as relevant as possible.
- Incorporating continuous integration and daily testing into the development process, allowing the development team to address issues while they're still fresh.
- Taking advantage of automated testing tools.
- Conducting sprint retrospectives, allowing the scrum team to continuously improve processes and work.
- Completing work using the definition of done: developed, tested, integrated, and documented.
- Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.

2. Higher customer satisfaction

The product owner is always involved, the progress of development has high visibility and flexibility to change is highly important. This implies engagement and customer satisfaction.

- Demonstrating working functionalities to customers in every sprint review.
- Delivering products to market quicker and more often with every release. The clients get early access to the product during the life cycle.
- Keeping customers involved and engaged throughout projects.

3. Increased project control

- Sprint meetings.
- Transparency.
- Jira usage (visibility of each step of the project for both parties).

4. Reduced risks

- Agile methodologies virtually eliminate the chances of absolute project failure.
- Always having a working product, starting with the very first sprint, so that no agile project fails completely.
- Developing in sprints, ensuring a short time between initial project investment and either failing fast or knowing that a product or an approach will work.
- Generating revenue early with self-funding projects, allowing organisations to pay for a project with little up-front expense.
- Agile gives freedom when new changes need to be implemented. They can be implemented at very little cost because of the frequency of new increments that are produced.
- Adaptation to the client's needs and preferences through the development process. Agile commonly uses user stories with business-focused acceptance criteria to define product features. By focusing features on the needs of real customers, each feature incrementally delivers value, not just an IT component. This also provides the opportunity to beta test software after each iteration, gaining valuable feedback early in the project and providing the ability to make changes as needed.

5. Faster ROI

The fact that agile development is iterative means that the features are delivered incrementally, therefore benefits are realised early while the product is in development process.

- Development starts early.
- A functional 'ready to market' product after few iterations.
- First Mover Advantage.
- Long delivery cycles are often a problem for businesses, particularly those in fast-moving markets.
- Agile means fast product releases and ability to gauge customer reaction and alter accordingly, keeping you ahead of the competition.
- Focusing on Business value. By allowing the client to determine the priority of features, the team understands what's most important to the client's business, and can deliver features in the most valuable order.

Benefits of Agile Methodology

With Agile project management, primary constraints, such as time and cost, can be continuously evaluated. Rapid feedback, continuous adaptation and Q&A best practices are built into teams' schedules, which ensures quality output and a streamlined process. The following represents some of the other benefits of Agile project management:

- Increased focus on the specific needs of customers
- Reduced waste through minimizing resources
- Increased flexibility enabling teams to easily adapt to change
- Better control of projects
- Faster project turnaround times
- Faster detection of product issues or defects
- Increased frequency of collaboration and feedback
- Improved development process
- Increased success because efforts are more focused

- Rapid deployment of solutions