**MCA (Management)**

**(2020-22)**

**Course Code:- IT 42**

**Course Name :- Software Project Management**

**Chapter 5 :- Tracking Agile Project and Reports**

* **5.1 Introduction :-**

One tool of an agile management system is the sprint backlog, which is a list of requirements and tasks in a given sprint. The sprint backlog is updated every day of the sprint. Make the sprint backlog available to the entire project team every day. That way, anyone who needs to know the sprint status can find it instantly.

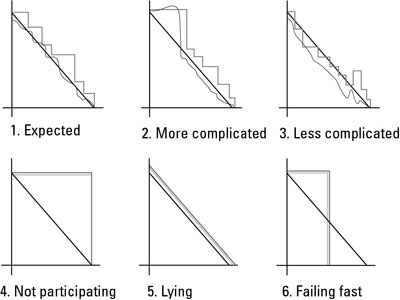
A *sprint burndown chart* (sample shown) shows the progress the development team is making and is a powerful tool for visualizing progress and the work remaining. The chart shows:

* The outstanding work (in hours) on the first vertical axis
* Time, in days, along the horizontal axis

Some sprint burndown charts also show the outstanding story points on a second vertical axis plotted against the same horizontal time axis as hours of work remaining.

A burndown chart enables anyone to see the status of the sprint at any time. Progress is clear. By comparing the realistic number of hours available to the actual work remaining, you can find out daily whether the effort is going as planned, is in better shape than expected, or is in trouble. That information helps you determine whether the development team is likely to accomplish the targeted number of user stories and helps you make informed decisions early in the sprint.

Looking at samples of burndown charts for sprints in different situations, you can tell how the work is progressing:



1. **Expected**

This chart shows a normal sprint pattern. The remaining work hours rise and fall as the development team completes tasks, ferrets out details, and identifies tactical work it may not have initially considered. Although work occasionally increases, it is manageable, and the team mobilizes to complete all user stories by the end of the sprint.

1. **More complicated**

In this sprint, the work increased beyond the point in which the development team felt it could accomplish everything. The team identified this issue early, worked with the product owner to remove some user stories, and still achieved the sprint goal. The key to scope changes within a sprint is that they are always initiated by the development team — no one else.

1. **Less complicated**

In this sprint, the development team completed some critical user stories faster than anticipated and worked with the product owner to identify additional user stories it could add to the sprint.

1. **Not participating**

A straight line in a burndown means that the team didn’t update the burndown or made zero progress that day. Either case is a red flag for future problems.

1. **Lying (or conforming)**

This burndown pattern is common for new agile development teams used to reporting the hours management expects instead of the time the work really takes. A team with this chart likely adjusted its work estimates to the exact number of remaining hours. This pattern often reflects a fear-based environment, where the managers lead by intimidation.

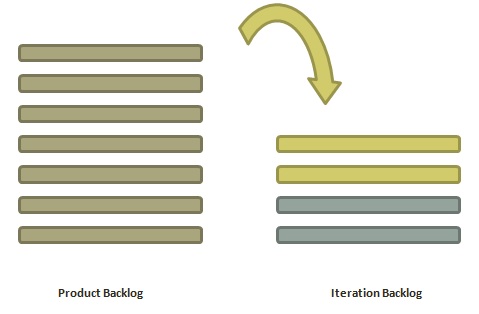
1. **Failing fast**

One of the strongest benefits of agile is the immediate proof of progress, or lack thereof. This pattern shows an example of a team that wasn’t participating or progressing. Halfway through the sprint, the product owners cut their losses and killed the sprint. Only product owners can end a sprint early.

The sprint backlog helps you track progress throughout each sprint. You can also refer to earlier sprint backlogs to compare progress from sprint to sprint. You will make changes to your process in each sprint. Constantly inspect your work and adapt to make it better.

* **5.2 Plan & Execute Iteration :-**

The purpose of iteration planning is for the team to complete the set of top-ranked product backlog items. This commitment is time boxed based on the length of iteration and team velocity.



**Who is Involved?**

* **Scrum Master** − The scrum master acts as a facilitator for the agile delivery team.
* **Product Owner** − The product owner deals with the detailed view of the product backlog and their acceptance criteria.
* **Agile Team** − Agile delivery defines their tasks and sets the effort estimates required to fulfil the commitment.

**Prerequisites of Planning**

* Items in product backlog are sized and have a relative story point assigned.
* Ranking has been given to portfolio items by the product owner.
* Acceptance criteria has been clearly stated for each portfolio item.

**Planning Process**

**Following are the steps involved in iteration planning –**

* Determine how many stories can fit in an iteration.
* Break these stories into tasks and assign each task to their owners.
* Each task is given estimates in hours.
* These estimates help team members to check how many task hours each member have for the iteration.
* Team members are assigned tasks considering their velocity or capacity so that they are not overburdened.

**Velocity Calculation**

An agile team calculates velocity based on past iterations. Velocity is an average number of units required to finish user stories in an iteration. For example, if a team took 12, 14, 10 story points in each iteration for the last three iterations, the team can take 12 as velocity for the next iteration.

Planned velocity tells the team how many user stories can be completed in the current iteration. If the team quickly finishes the tasks assigned, then more user stories can be pulled in. Otherwise, stories can be moved out too to the next iteration.

**Task Capacity**

The capacity of a team is derived from the following three facts −

* Number of ideal working hours in a day
* Available days of person in the iteration
* Percentage of time a member is exclusively available for the team.

Suppose a team has 5 members, committed to work full time (8 hours a day) on a project and no one is on leave during an iteration, then the task capacity for a two-week iteration will be −

5 × 8 × 10 = 400 hours

**Planning Steps**

* Product Owner describes the highest ranked item of product backlog.
* Team describes the tasks required to complete the item.
* Team members own the tasks.
* Team members estimate the time to finish each task.
* These steps are repeated for all the items in the iteration.
* If any individual is overloaded with tasks, then his/her task is distributed among other team members.

Iteration Planning is an event where all team members determine how much of the Team Backlog they can commit to delivering during an upcoming Iteration.  
  
**The purpose of iteration planning** is to organize the work and define a realistic scope for the iteration. Each Agile Team agrees on a set of stories for the upcoming iteration (the iteration backlog) and summarizes those stories into a set of iteration goals. The iteration backlog and goals are based on the team’s capacity and allow for consideration of each story’s complexity, size, and dependencies on other stories and other teams. At the end of planning, the teams commit to the goals of the iteration and adjust stories as necessary to achieve the larger purpose. In return, management does not interfere or adjust the scope of the iteration, allowing the team to stay focused.

**Number of inputs to the iteration planning event:** The Team and Program Increment (PI) Objectives, created during PI planning The team’s PI plan, which consists of stories that were identified during PI planning and provisionally assigned to iterations Additional stories that have arisen from a local context, including items such as defects and Re-factors, as well as new stories that have come about since the planning session to support Feature delivery Feedback from the prior iteration, including any stories that were not successfully completed (e.g., did not meet the ‘definition of done’) in that iteration. For more information on the definition of done, please see the ‘Scaled Definition of Done’ paragraph in the Built-in-Quality article. Feedback from the system demo Planning the Iteration Prior to iteration planning, the Product Owner (PO) will have prepared some preliminary iteration goals, based on the team’s progress in the Program Increment (PI) so far. Typically, the Product Owner starts the event by reviewing the proposed iteration goals and the higher-priority stories in the team backlog. During iteration planning, the Agile team discusses implementation options, technical issues, Nonfunctional Requirements (NFRs), and dependencies, then plans the iteration. The Product Owner defines the ‘what’; the team defines ‘how’ and ‘how much’. Throughout iteration planning, the team elaborates the acceptance criteria for each story and estimates the effort to complete each one. Based on their available capacity for the iteration, the team then selects the candidate stories. Some teams break each story down into tasks and forecast them in hours to confirm that they have the capacity and skills to complete them. Once this process is complete, the team commits to the work and records the iteration backlog in a visible place, such as a storyboard, Kanban board or Agile project management tooling. Planning is timeboxed to a maximum of four hours for a two-week iteration.

The team pulls the stories into the sprint backlog from the product backlog and groups them into independent tasks of fewer 8 hours each.

* They also perform risk assessment of stories and decides on a light weight plan on how to address the risks as the sprint moves forward.
* The team asks questions to the Product Owner on clarifications that might make them to understand the stories in more realistic way.
* The number of stories that the team pulls in the sprint may depend on team’s capacity or velocity (if known).
* The capacity planning is done in hours and the velocity planning is done in story points.
* The team makes a commitment to the sprint backlog and changes the sprint backlog as it emerges.

The testers are also involved in the iteration and can contribute to the same in the following ways:

* Breaking user stories into testing tasks
* Determining test coverage of every story
* Creating acceptance tests for user stories
* Estimate the testing tasks like creating test strategy, test plans and test cases specific to user stories
* Understand and plan for automating user stories and support various levels of testing
* **5.3 Facilitate Retrospective, Making team Decision and closing out Retrospective:-**

The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness.

The Scrum Team inspects how the last Sprint went with regards to individuals, interactions, processes, tools, and their Definition of Done. Inspected elements often vary with the domain of work. Assumptions that led them astray are identified and their origins explored. The Scrum Team discusses what went well during the Sprint, what problems it encountered, and how those problems were (or were not) solved.

The Scrum Team identifies the most helpful changes to improve its effectiveness. The most impactful improvements are addressed as soon as possible. They may even be added to the [Sprint Backlog](https://www.scrum.org/resources/what-is-a-sprint-backlog) for the next [Sprint](https://www.scrum.org/resources/what-is-a-sprint-in-scrum).

The Sprint Retrospective concludes the Sprint. It is timeboxed to a maximum of three hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.

During the Sprint Retrospective, the team discusses:

* What went well in the Sprint
* What could be improved
* What will we commit to improve in the next Sprint

The [Scrum Master](https://www.scrum.org/resources/what-is-a-scrum-master) encourages the rest of the Scrum Team to improve its process and practices to make it more effective and enjoyable for the next Sprint. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of “Done” if appropriate and not in conflict with product or organizational standards.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

An Agile retrospective is a meeting that's held at the end of an [iteration](https://searchsoftwarequality.techtarget.com/definition/iteration) in Agile software development ([ASD](https://searchsoftwarequality.techtarget.com/definition/agile-software-development) ). During the retrospective, the team reflects on what happened in the iteration and identifies actions for improvement going forward.

Each member of the team members answers the following questions:

* What worked well for us?
* What did not work well for us?
* What actions can we take to improve our process going forward?

The Agile retrospective can be thought of as a "lessons learned" meeting. The team reflects on how everything went and then decides what [changes they want](https://searchsoftwarequality.techtarget.com/tip/Six-tips-for-successful-change-management-in-an-Agile-environment) to make in the next iteration. The retrospective is team-driven, and team members should decide together how the meetings will be run and how decisions will be made about improvements.

Because Agile stresses the importance of [continuous improvement](https://searcherp.techtarget.com/definition/kaizen-or-continuous-improvement), having a regular Agile retrospective is one of the most important of Agile development practices. The Ninth Agile principle outlined in the [Agile manifesto](https://searchcio.techtarget.com/definition/Agile-Manifesto) states, "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." A framework, such as the one below, can be used to provide structure and keep discussion during the retrospective focused.

1. Set the stage - get the team ready to engage in the retrospective, perhaps with a warm-up activity such as Plus, Minus, Interesting
2. Gather data - create a shared picture of what happened during the retrospective
3. Generate insights - discuss what was successful and identify any roadblocks to success
4. Decide what to do - identify highest priority items to work on and put measurable goals on those items so they can be completed
5. Close the retrospective - reflect on the retrospective and how to improve it, and to appreciate accomplishments of the team and individual interactions

Agile teams use retrospectives to reflect, learn, and change their way of working. The facilitator of the retrospective plays an important role to make retrospectives effective. Let's explore what facilitators can do to help teams to get more value out of their retrospectives.

An [agile retrospective](http://www.benlinders.com/2013/whats-an-agile-retrospective-and-why-would-you-do-it/), or sprint retrospective as Scrum calls it, is a practice used by teams to reflect on their way of working, and to continuously become better in what they do.

The whole team attends the retrospective meeting, where they “inspect” how the iteration (sprint) has been done, and decide what and how they want to “adapt” their [processes](http://www.benlinders.com/2010/what-is-a-%e2%80%9cprocess%e2%80%9d/) to improve.  The actions coming out of a retrospective are communicated and done in the next iteration. That makes retrospectives an effective way to do [short cycled improvement](http://www.benlinders.com/2013/short-cycled-improvement-observe-and-change/).

**Role of the facilitator**

The retrospective facilitator has to do everything that is needed to make sure that the retrospective is valuable for the team. This includes:

* Planning the meeting and assuring that team members can attend the retrospective
* Selecting appropriate exercises and preparing them
* Running the retrospective meeting
* Setting the conditions for effective follow up on the actions

To make the retrospective effective, the facilitator shall:

* Establishing a open and honest culture in the meeting
* Ensure that all team members participate in the meeting
* Assure that the team establishes a shared understanding of how things went
* Help the team to decide upon the vital few actions that they will take

Here's a couple of practices that recommend for facilitating retrospectives to make them valuable for the participants.

Many retrospective facilitators use the **prime directive** to establish a culture where team members speak up and will be open and honest. It sets the assumption that team member did the best they could possibly do, and that the purpose of the retrospective is not to blame people.

A retrospective facilitator should **not have a personal stake in the outcome** of the meeting. The team has to decide which actions they will do, the facilitator should not influence their decision. If the Scrum master is facilitating the retrospective this can be a challenge, as He or She is also a team member.

Retrospective facilitators must be able to **deal with negative issues**. They should help the team focus on the issue and to understand them and don't blame any team members for what has happened. This requires strong communication skills, paying attention to verbal and non verbal communication.

Dealing with **silence** is another skill that facilitators should have. Silence helps people to think, to reflect, or to accept things that have happened. Sometimes the best thing a facilitator can do is to say and do nothing and give space to the team members to come up with their ideas.

A facilitator should **focus on the process** of the retrospective meeting. They should assure that exercises are done in an appropriate way, keep time, and help team members in performing activities that are needed to do the retrospective.

A facilitator should **serve their the team** in the meeting, but should not lead them based on their own opinion on what the team should do. It is important that a facilitator remains independent, which again can be challenging when the facilitator is also the Scrum master of the team.

**Active listening** can be useful in retrospective meetings, for instance when the facilitator recaps what is being said and checks if it has been understood by all attendants.

**one-word exercise example :-**

* Establish trust and openness
* Respect people and their feeling
* Deal with the issues

Trust is important in any retrospective, even more when you are dealing with people’s feelings and emotions. The team members have to feel safe to be open; as a facilitator you have to make clear that what is being said will remain within the team. It is also up to the team what they want to do with it, even if they decide that they do not want to take action.

As a facilitator you have to respect the opinions of the team members, and also assure that they respect each other. If people start blaming or accusing, please remind them that the purpose of a retrospective is to understand what happened and learn from it. Remember them of the retrospective prime directive.

It’s important to deal with the issues that are brought up. People reach out, and take risks by discussing them, they have to feel rewarded by the fact that something is done by team with it. The team members have to leave the room feeling that they have been heard and understood, and have the feeling that they can solve the issues as a team.

**To get feedback with a perfection game** you ask people to provide answers to the following questions:

* I rate the product/service … on a scale from 1-10
* What I liked about it …
* To make it perfect …

People have to rate the value that they received on a scale from 1 to 10, based on how much value they think they could add themselves by improving the product or service. For example, when there is nothing that they think they can improve, they should rate with a 10. If they think that they could make it twice as valuable, they should give it a 5.

**Better Feedback**

In the perfection game, the rating is coupled to what people like about it and what they think can be done to do it better. This improves the quality of they feedback as people have to motivate their ratings.

If somebody rates the value as 5, He or She will have to provide suggestions how to make it a 10. People can only withhold points if they provide suggestions to improve it.

If people give a high rating then they have to state what they like about it. What makes it so good, where does the value come from.

**How to do a futurespective**

In a futurespective teams places themselves in the future by imagining that their goal has been reached. They start such a retrospectives by discussing the team goals to ensure that team members build a common understanding.  The goals are formulated and written down so that they are visible for everybody.

Optionally teams can write down which benefits they got from reaching their goals. If teams likes to party they can even do a small celebration for having reached the goals, which can help to make teams aware of the importance of reaching it.

Next teams discuss their imaginary past and explore how they have gotten to their goals. There are two things that that teams question themselves:

* What are the things that have helped us to get here?
* Which things made it hard for us to reach our goal?

The 12th principle of the [Agile Manifesto](https://agilemanifesto.org/principles.html) states: “**At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly**”.

A sprint retrospective, also known as an agile retrospective, is a meeting held by a team at the end of a sprint to review its process and identify opportunities to improve it.

The purpose of a retrospective is to:

* Inspect how the last sprint went with regards to people, relationships, process, and tools;
* Identify and order the major items that went well and potential improvements;
* Create a plan for implementing improvements to the way the scrum team does its work.

To put it simply, a sprint retrospective should **create a safe space for people to share their honest feedback** on what's going well and what isn't, and to **generate a discussion around what could be improved next time**. It's a practice that helps teams reflect on their way of working and continuously become better in what they do.

**Sprint retrospective Vs. Sprint review :-**

Sprint retrospectives are often confused with sprint reviews, but they are not the same.

A sprint review takes place before the sprint retrospective and is used as an opportunity to discuss what has been accomplished during the sprint and whether the sprint goal has been met. It is used to review and plan ways to improve the product, while the sprint retrospective is used to review and improve the processes used to create the product.

Essentially, the *sprint review* is a discussion about **what the team is building**, while the *retrospective* is focused on **how they’re building it**.

* What did we do right in the previous sprint?
* What did we do wrong in the previous sprint?
* What should we start doing in the next sprint?
* What should we stop doing in the next sprint?
* What can we do to improve productivity?

**Every team should be conducting two forms of Retrospectives:**

1. **The Process Retrospective (aka Post-mortem)**: At the end of every project sprint or deliverable, the project team involved should look back, create a timeline of the work, and reflect on specific points along the way that were either positive or negative to the outcome of the work.
2. **The Team Retrospective**: At the end of every month, each department should reflect on its rules, habits, and processes to identify better ways to work together.

To Improve Project Process :-

What factors have contributed to successes so far?

Which tough spots could have been handled differently?

What opportunities and/or challenges should you anticipate as you move forward?

* Analyze findings.
* Present and discuss findings—successes, opportunities to improve, and recommendations for next steps.
* Prepare a final summary or report.
* Determine actions.

The retrospective is a critical part of **closing out a project**. During the retrospective, the project team comes back together to reflect on the project experience and the final results. **Together, they identify what went well, what didn’t, and where improvements can be made for next time.**

1. Sum up the results (Meeting Recap)
2. Action Items
3. Follow-ups
4. Perform a Retrospective of the Retrospective session
5. Thank everyone and let them go

You can ask everyone to name one thing what they learned in this Retrospective. Then you give the word to each participant in the group.

Another option is to ask everyone to write down on a sticky note one thing they like and one thing they would change about the Retrospective. Then everyone, one after another, puts the stickier on the board explaining what they mean.

**Why do a closeout?**

It’s important to do an effective close for several reasons

* It can help ensure that everyone is as happy as can be with the work done — enabling everyone to know where things stand
* It helps draw a line under the work & lets people move on emotionally
* It increases the impact of the team’s work by, e.g. communicating and sharing it more effectively
* It helps individuals from the team and those working to support the team do better next time
* It helps anyone picking up the team’s deliverables or ways of working

**Learning Lesson :-**

* Over the course of the project what were the important lessons learned about how the team worked?
* How effective were you at working in an agile way? How might you have organized yourself differently?
* What could you improve in the environment around the delivery team — e.g. IT provisioning, project governance, your team space , were stakeholders engaged?
* If you used suppliers, how did they do? Did you spend money wisely? Would **you** use them again? Should **your organization** use them again?
* What was your own personal contribution? Take time to celebrate your own successes. Solicit feedback from others on your strengths & what you might take forward for your next piece of work.
* **5.4 Agile Reports :-**

Reporting is the direct result of the inherent need to measure, digest, and understand key data for decision making. In Agile, that reporting part must be quick and easy to get, read, and understand. Add to this the fact that in Agile you do not have time to build reporting during the execution of any iteration.

* **5.4.1 Daily Reports :-**

Daily stand-up, as the name suggests, is a daily status meeting among all the members of an agile team. It not only provides a forum for regular updates but also brings the problems of team members into focus so that it can be quickly addressed. Daily stand-up is a must-do practice, no matter how an agile team is established regardless of its office location.

**What is Daily Stand-up?**

* A daily stand-up is a daily status meeting among all team members and it is held roughly for 15 minutes.
* Every member has to answer three important questions −
  + What I did yesterday?
  + What I'll do today?
  + Any impediment I am facing.../ I am blocked due to...
* Daily stand-up is for status update, not for any discussion. For discussion, team members should schedule another meeting at a different time.
* Participants usually stand instead of sitting so that the meeting gets over quickly.

**Why Stand-up is Important?**

The benefits of having a daily stand-up in agile are as follows −

* The team can evaluate the progress on a daily basis and see if they can deliver as per the iteration plan.
* Each team member informs all about his/ her commitments for the day.
* It provides visibility to the team on any delay or obstacles.

**Who Attends a Stand-up?**

* The scrum master, the product owner, and the delivery team should attend the stand-up on a daily basis.
* Stakeholders and Customers are encouraged to attend the meeting and they can act as an observer, but they are not supposed to participate in stand-ups.
* It is the scrum master's responsibility to take note of each team member's queries and the problems they are facing.

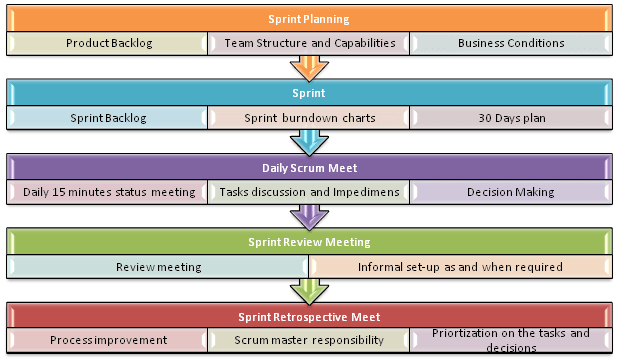
**Geographically Dispersed Teams :-**

Stand-ups can be done in multiple ways, in case the agile team members are operating from different time zones −

* Select a member on a rotational basis, who can attend the stand-up meeting of teams located in different time zones.
* Have a separate stand-up per team, update the status of the stand-up in a tool such as Rally, SharePoint, Wikis, etc.
* Have a wide variety of communication tools ready like conference call, video conferencing, instant messengers, or any other third-party knowledge sharing tools.

**Scrum Practices**

Practices are described in detailed:

[](https://cdn.guru99.com/images/11-2014/agile_Processesv1_4.png)

**Process flow of Scrum Methodologies:**

Process flow of scrum testing is as follows:

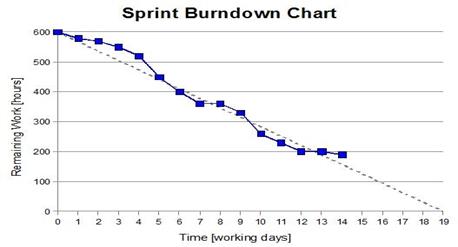
* Each iteration of a scrum is known as Sprint
* Product backlog is a list where all details are entered to get the end-product
* During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
* Team works on the defined sprint backlog
* Team checks for the daily work
* At the end of the sprint, team delivers product functionality
* **5.4.2 Sprint Burn down Chart & Reports :-**

**In year 2000**: The burn down chart is first described by Ken Schwaber, who invents it while working at Fidelity Investments in an [attempt](https://groups.yahoo.com/neo/groups/scrumdevelopment/conversations/messages/16) to provide Scrum teams with a simple tool kit; he describes it formally [on his Web site](http://web.archive.org/web/20010503112119/www.controlchaos.com/sburndown.htm)

**In year 2002**: The burn down gains popularity among the Scrum community, as well as alternatives such as the “burn up” which merely inverts the vertical direction

**Burn down chart:** Each day, Scrum Master records the estimated remaining work for the sprint. This is nothing but the Burn Down Chart. It is updated daily.

A burn down chart gives a quick overview of the project progress, this chart contains information like the total amount of work in the project that must be completed, amount of work completed during each sprint and so on.

[](https://cdn.guru99.com/images/11-2014/112714_1232_ScrumTestin4.jpg)

**Velocity history graph:** The velocity history graph predicts the velocity of the team reached in each sprint. It is a bar graph and represents how teams output has changed over time.

The additional metrics that may be useful are schedule burn, budget burn, theme percent complete, stories completed - stories remaining and so on.

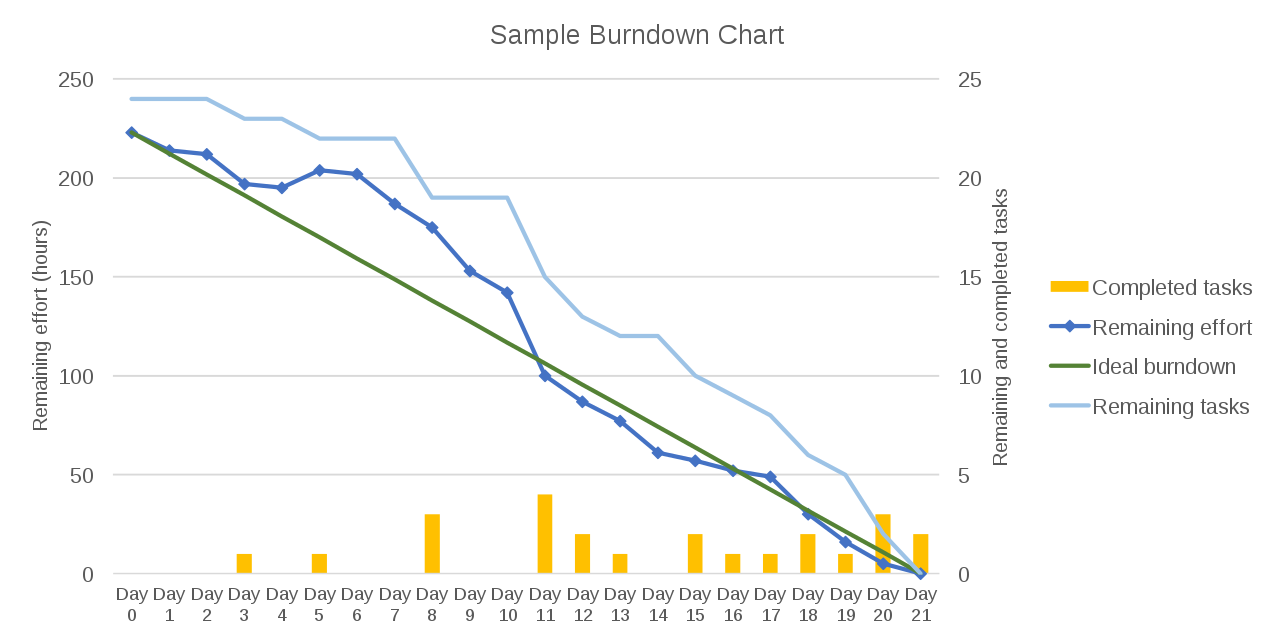
Burn Down Chart is a large graph relating the quantity of work remaining (on the vertical axis) and the time elapsed since the start of the project (on the horizontal, showing future as well as past). This constitutes an “[information radiator](https://www.agilealliance.org/glossary/information-radiators/)“, provided it is updated regularly. Two variants exist, depending on whether the amount graphed is for the work remaining in the iteration (“sprint burn down”) or more commonly the entire project (“product burn down”).

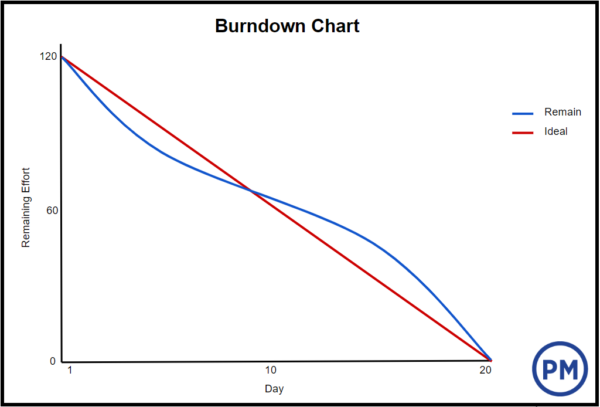
Burn down charts only show the number of story points completed, they do not indicate any changes in the scope of work as measured by total points in the backlog.  As a result, it’s difficult to tell whether changes in the burndown chart can be attributed to backlog items completed, or simply and increase (or much less likely) a decrease in story points. The burn up chart resolves this issue by showing a separate line for overall backlog size.

Neither the burn down or burn up chart provides any indication of which product backlog items have been completed. This means that a team can have a burn down chart that shows continued progress, but it does not indicate whether the team is working on the correct things.  For this reason, burn down and burn up charts can only provide an indication of trends rather than giving explicit indication of whether a team is delivering the right product backlog items.

A **burn down chart** is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. Burn down charts are a [run chart](https://en.wikipedia.org/wiki/Run_chart) of outstanding work. It is useful for predicting when all of the work will be completed. It is often used in [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) methodologies such as [Scrum](https://en.wikipedia.org/wiki/Scrum_(development)). However, burn down charts can be applied to any project containing measurable progress over time.

Outstanding work can be represented in terms of either time or story points.





**Reading burn down charts:-**

[](https://en.wikipedia.org/wiki/File:Burn_down_chart.png)

**A project burn down chart :-**

A burn down chart for a completed iteration is shown above and can be read by knowing the following:

|  |  |
| --- | --- |
| X-Axis | The project/iteration timeline |
| Y-Axis | The work that needs to be completed for the project. The time or story point estimates for the work remaining will be represented by this axis. |
| Project Start Point | This is the farthest point to the left of the chart and occurs at day 0 of the project/iteration. |
| Project End Point | This is the point that is farthest to the right of the chart and occurs on the predicted last day of the project/iteration |
| Number of Workers and Efficiency Factor | In the above example, there are an estimated 28 days of work to be done, and there are two developers working on the project, who work at an efficiency of 70%. Therefore, the work should be completed in (28 ÷ 2) ÷ 0.7 = 20 days. |
| Ideal Work Remaining Line | This is a straight line that connects the start point to the end point. At the start point, the ideal line shows the sum of the estimates for all the tasks (work) that needs to be completed. At the end point, the ideal line intercepts the x-axis showing that there is no work left to be completed. Some people take issue with calling this an "ideal" line, as it's not generally true that the goal is to follow this line. This line is a mathematical calculation based on estimates, and the estimates are more likely to be in error than the work. The goal of a burn down chart is to display the progress toward completion and give an estimate on the likelihood of timely completion. |
| Actual Work Remaining Line | This shows the actual work remaining. At the start point, the actual work remaining is the same as the ideal work remaining but as time progresses, the actual work line fluctuates above and below the ideal line depending on this disparity between estimates and how effective the team is. In general, a new point is added to this line each day of the project. Each day, the sum of the time or story point estimates for work that was recently completed is subtracted from the last point in the line to determine the next point. |

**Measuring performance :-**

|  |  |
| --- | --- |
| Actual Work Line is above the Ideal Work Line | If the actual work line is above the ideal work line, it means that there is more work left than originally predicted and the project is behind schedule. |
| Actual Work Line is below the Ideal Work Line | If the actual work line is below the ideal work line, it means that there is less work left than originally predicted and the project is ahead of schedule. |

The above table is only one way of interpreting the shape of the burn down chart. There are others.

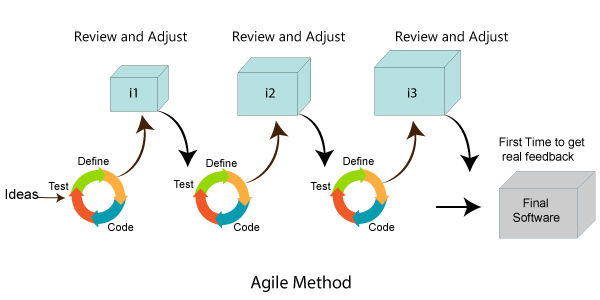
**Removing variability in time estimates :-**

One issue that may be noticed in burn down charts is that whether or not the Actual Work line is above or below the Ideal Work line depends on how accurate the original time estimates are. This means that if a team constantly overestimates time requirements, the progress will always appear ahead of schedule. If they constantly underestimate time requirements, they will always appear behind schedule. This issue is corrected by incorporating an efficiency factor into the burn down chart. After the first iteration of a project, the efficiency factor can be recalculated to allow for more accurate estimates during the next iteration. Some templates automatically calculate the efficiency as a project progresses. This can be used to identify areas/phases where inaccurate estimates consistently occur

* **5.5 Benefits of Agile Project Management :-**

**Advantages of Agile Methodology**

1. Customer satisfaction is rapid, continuous development and delivery of useful software.
2. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
3. Product is developed fast and frequently delivered (weeks rather than months.)
4. A face-to-face conversation is the best form of communication.
5. It continuously gave attention to technical excellence and good design.
6. Daily and close cooperation between business people and developers.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed.
9. More Control , Better Quality , Better Productivity, Improve Predictability
10. Higher Return on Investment
11. Reduced Risk & Increased Flexibility
12. Continuous Improvement & Team Moral
13. More relevant Metric in terms of Time, Cost , Performance

****

**Disadvantages of Agile methodology:**

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

**When to use Agile model:**

* When new changes need to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
* To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
* Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users’ needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
* Both system developers and stakeholders alike find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.