PRODUCT MANAGEMENT

RELEASE PLANNING

- Release Planning happens throughout the Project.
- It provides a forecast of which functionality can be delivered by when to relevant stakeholders.

Decision on Time, Cost, and Functionality

- Release Planning starts with deciding what is more important for a successful product launch; whether time, cost, and functionality.

TIME

- Time or Product Launch date is a critical factor for Release Planning. The whole release mechanism and the corresponding activities are undertaken to ensure that this launch date is met.
- Product Launch is an opportunity for the organization. If the team misses the launch date, it is an opportunity lost.
- Fixing the launch date enables to fix the timeboxes for all releases – for example, quarterly release helps the organization to launch product increments every quarter.

COST

- Fixed Price contracts often freeze the time and cost—time and cost are fixed for the project. In such cases, scope may be fixed or a portion of scope is kept volatile.
- When cost is fixed, it poses significant challenges to supplier teams. If scope is not fixed—Agile is used.
- In most cases, organizations establish Change Budget to handle changes as they come in Fixed Price contracts.

FUNCTIONALITY

- Remember Agile Manifesto principle “Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”
- Keeping the functionality fixed is against Agile. Not recommended at all.
- Requirements emerge and Product Backlog evolves over a period of multiple releases.

Frozen Quality in Scrum

- Quality remains frozen in Agile software projects.
- By using a well-defined “Definition of Done” that is agreed between Product Owner and the team, the quality requirements are frozen.
- In typical waterfall projects, quality control and assurance might be carried out toward the end of the Project but in Agile quality control and assurance are carried out throughout the Project.
- Each sprint should deliver the Product increment with agreed and right quality.

- Product Owner is the ultimate decision maker on quality of User Stories and must not compromise on Definition of Done or User Stories that doesn't meet the Definition of Done.

Early and Frequent Releases

- Remember these two Agile Manifesto principles "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software" and "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."
- Early and frequent releases provide the opportunity to earn invaluable customer feedback early in Product development.
- It allows the product to evolve per customer feedback and responses.
- Frequent releases deploy the intended Product (or Increment) to Production Environment (the environment where the final product is supposed to operate). This creates an opportunity for customers to evaluate the Product. Compare this approach to testing on Quality Environments or using Prototypes—they may never match or replicate Production Environment.
- It is important that the resultant software must be of high quality. However, one concern with frequent releases is that with each release a set of functionality is released. If these sets of functionalities are not tested thoroughly (both for expected new functionality and integration with existing functionality), it can bring the complete product to a halt, thus impacting the brand value and customer loyalty.

Quarterly Cycles

- The release frequency depends on the product to be launched and customer requirements.
- However, in Agile projects, the best practice is to keep the release frequency between 3 and 6 months.
- If more than 3 to 4 months are required, then it is beneficial to use a quarterly release cycle.

Velocity

- Remember Agile Manifesto principle: "Working software is the primary measure of progress."
- Velocity ***is*** the indicator of how much work a Scrum team can do in a Sprint.
- It is used to track and forecast a project's progress.
- Velocity is the sum of story points of completed User Stories.
- The User Stories that meet the Definition of Done agreed between Product Owner and Development team are completed User Stories.

- Let us calculate Velocity with the following example. User Story IDs are for illustration purposes.
    - At Sprint Planning meeting, the following table shows selected User Stories for the Sprint:

User Story	Story Points
56345	1
78335	3
12349	3
43908	5

    - From the above data, **planned velocity is 1+3+3+5=12**.
    - During Sprint Review, the result of the selected User Stories and the status of meeting the Definition of Done are as follows:

User Story	Story Points	Definition of Done
56345	1	Met
78335	3	Met
12349	3	Not met
43908	5	Met

    - From above data, **the actual velocity is 1+3+5=9**. Three story points for Story# 12349 are not counted toward Velocity because this story has not met its Definition of Done.
- Velocity is specific to a team and ***can't be compared across teams.***

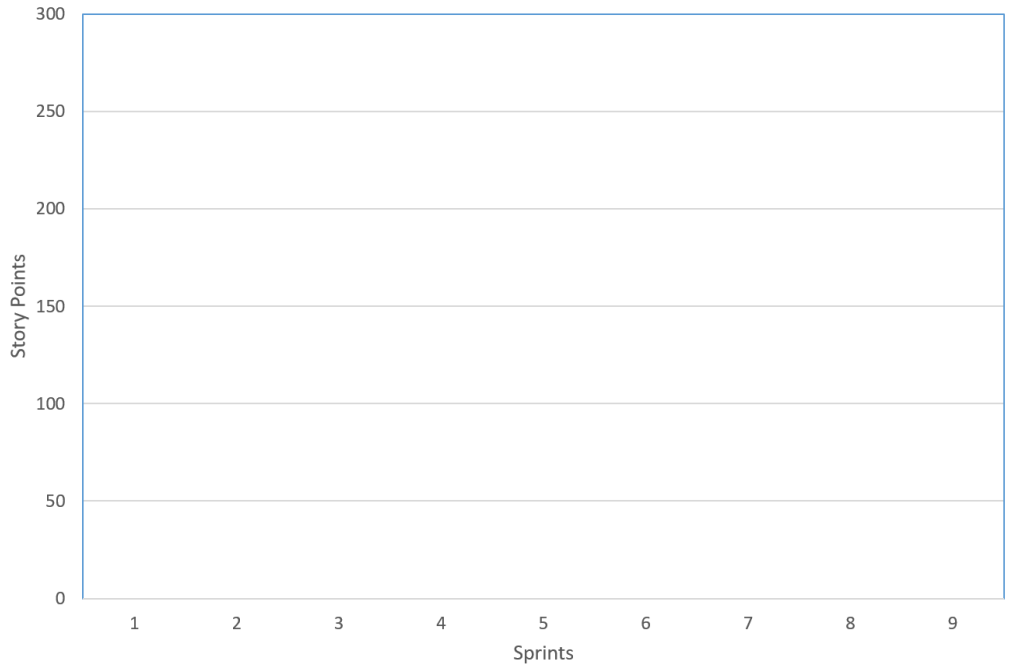
Release Burndown

- Release Burndown is one of the primary artifacts in Scrum to track and forecast project progress.
- It comes in two flavors:
    - Burndown Chart
    - Burndown Bar

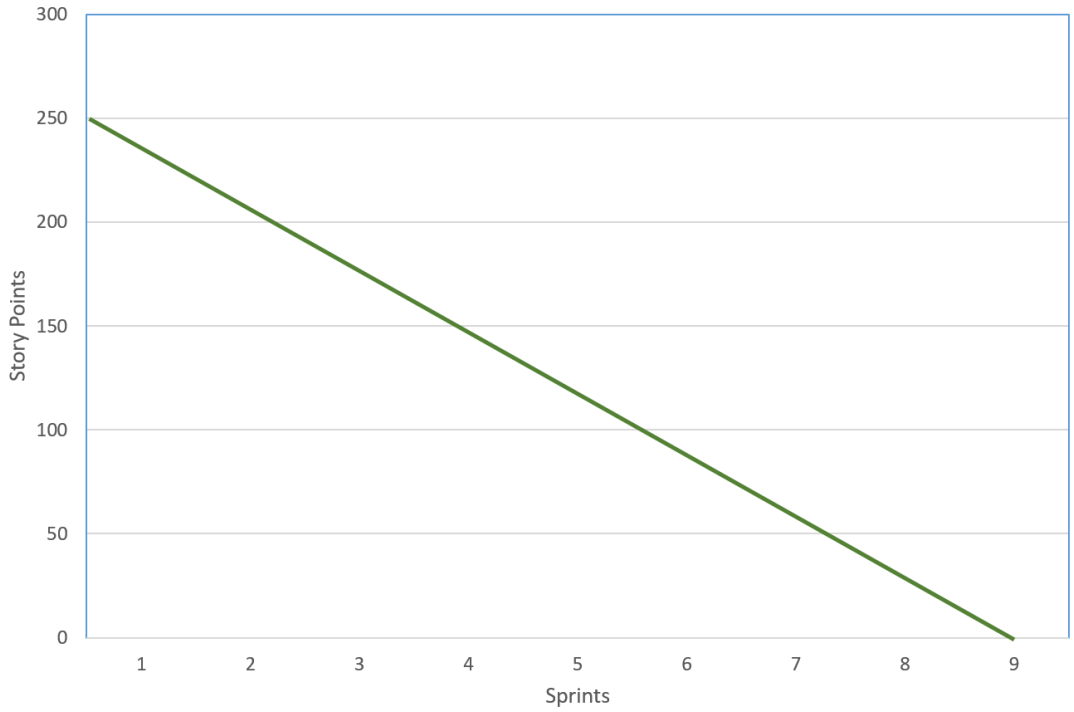
RELEASE BURNDOWN CHART

- Based on Velocity of previously completed Sprints, the release burndown helps teams to anticipate the future.
- It relies on two factors: remaining effort with respect to Product Backlog and time.
- Let us understand this with an illustrated example:
    - The plan is for 9 Sprints.

- Product Backlog contains User Stories with a total of 250 Story Points.
- On X-Axis (horizontal) is 9 Sprints, and on Y Axis is 300 Story Points.

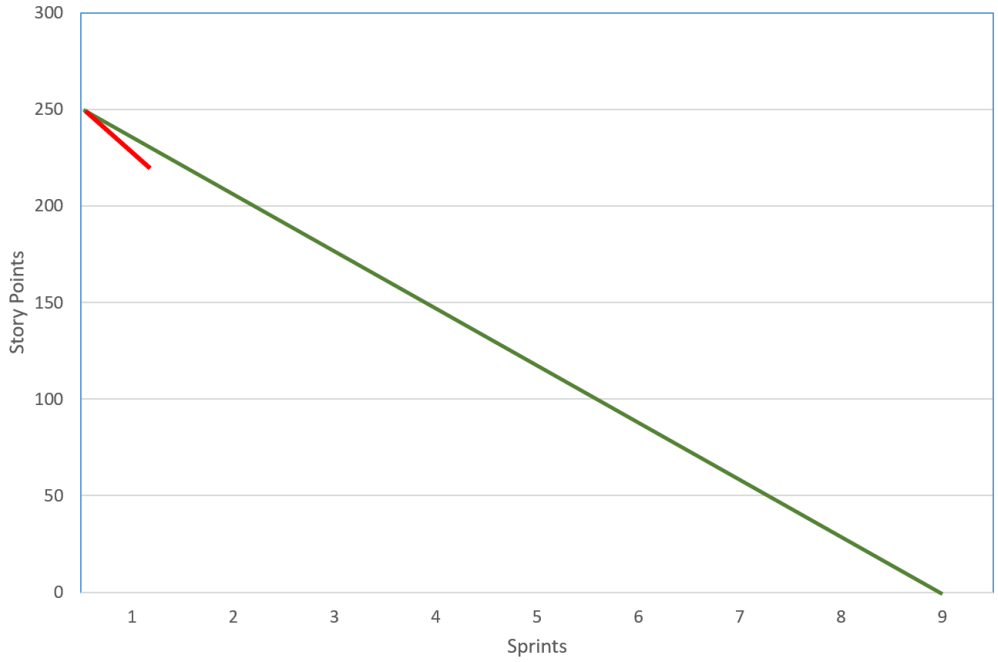


- The below illustration shows a ***forecast*** of how the team can meet the remaining effort or Story Points across the 9 Sprints. This is indicated by the green line.

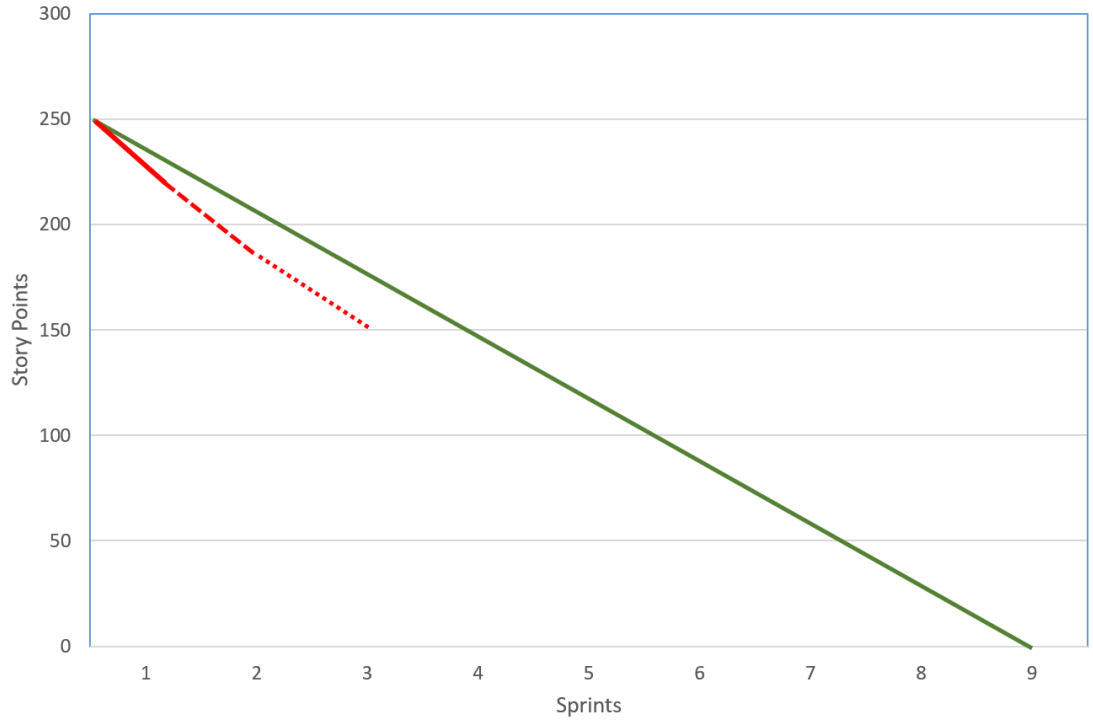


- **Situation 1:** Let us say, the team gets a velocity of 30 (remember: velocity is for completed User Stories per Definition of Done) for Sprint 1. This means remaining effort or Story

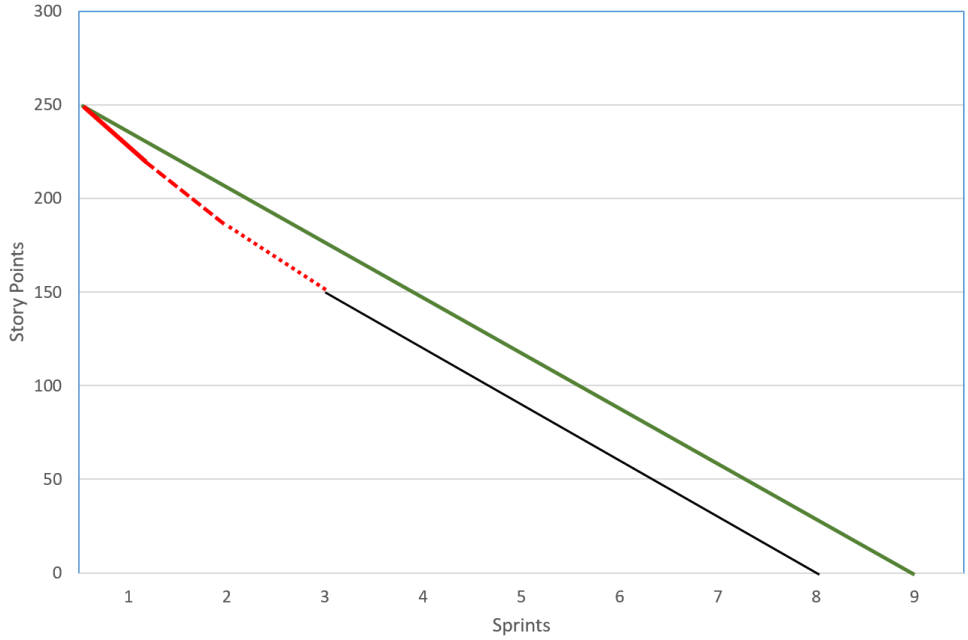
Points are 250 – 30 = 220. This is illustrated below with the red line.



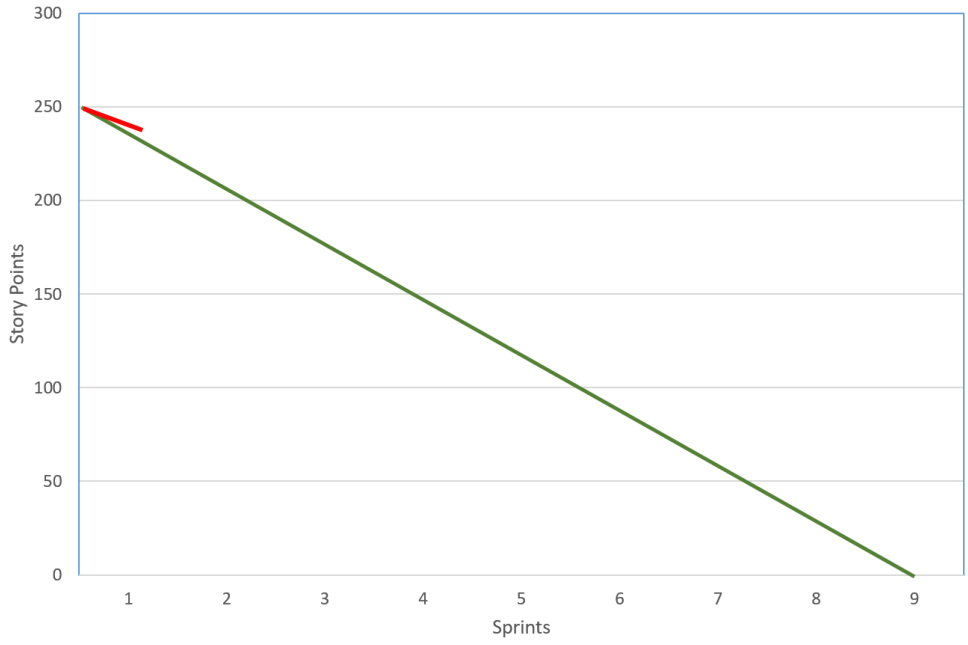
- In Sprint 2, the velocity is 40 (remaining Story Points will be 220 – 40 = 180). In Sprint 3, the velocity is 30 (remaining Story Points will be 180 – 30 = 150). This is illustrated below with the red line. These red lines are shown dotted to distinguish from the first one.



- ***It is evident from this trend that if the team continues this way, they may complete the project before the ninth sprint.*** This is illustrated below using the black line.

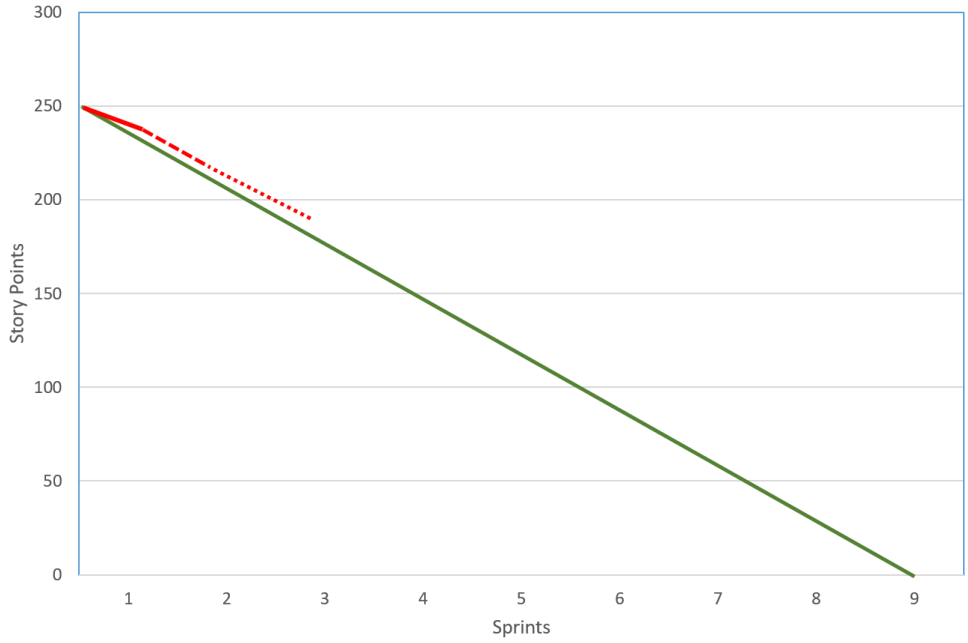


- **Situation 2:** Let us assume the team gets a velocity of 10 for Sprint 1. The remaining effort or Story Points are 250 – 10 = 240. This is illustrated below with the red line.

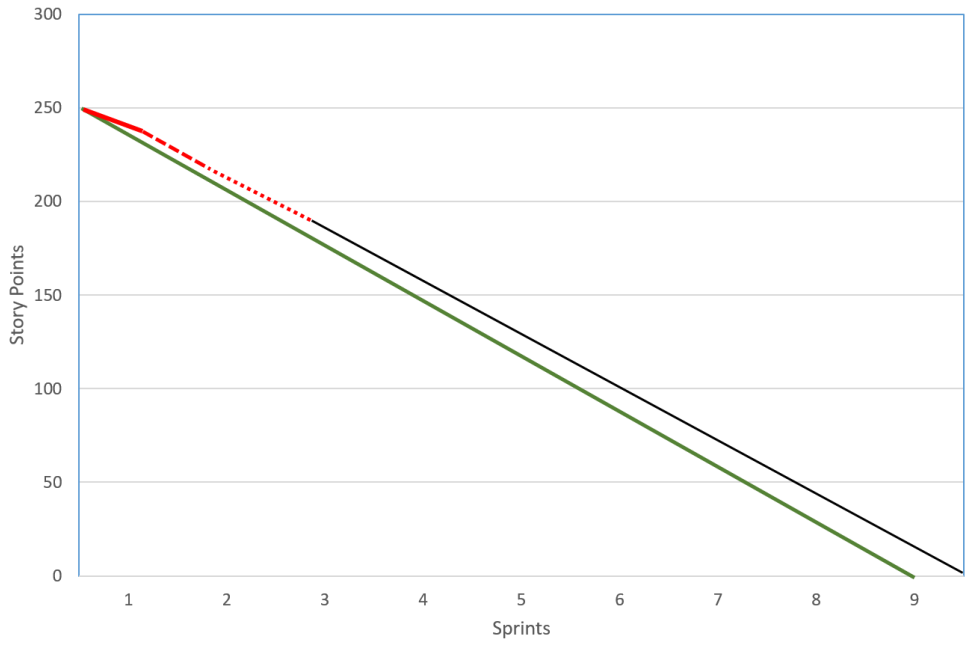


- Let us assume velocity of Sprint 2 is 20 (remaining Story Points are 240 – 20 = 220) and Sprint 3 is 30 (remaining Story Points are 220 – 30 = 190). This is illustrated below with the

red line. These red lines are shown dotted to distinguish from the first one.



- ***It is evident from this trend that if the team continues this way, they may complete the project after the ninth Sprint, which is behind schedule.*** This is illustrated below using the black line.

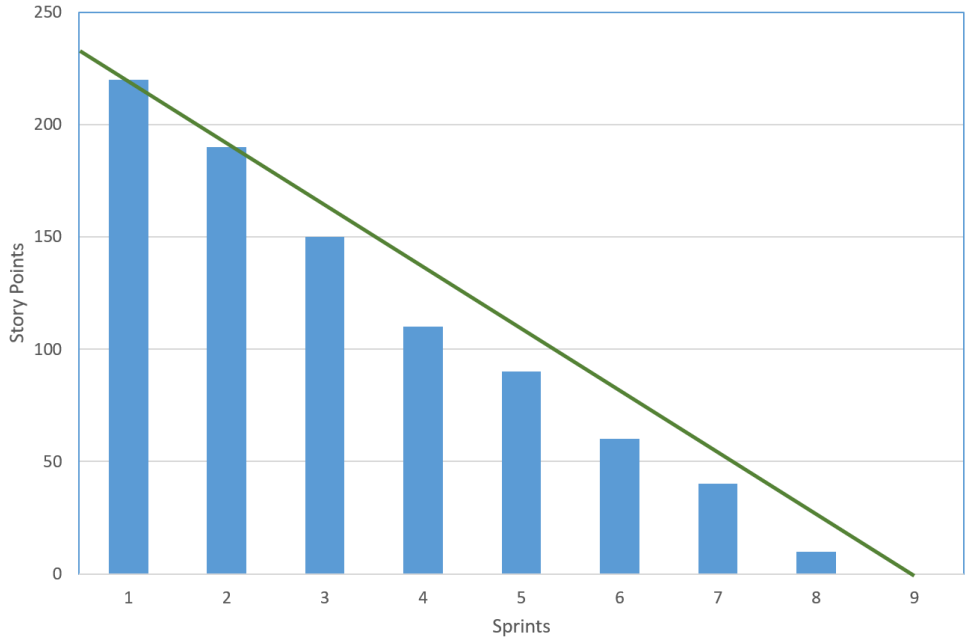


RELEASE BURNDOWN BAR

- This is similar to the Release Burndown chart except that it allows tracking re-estimation effort and adding/removing Backlog items.
    - If the team makes progress or reduces its estimates, the top of the bar moves down.
    - If the team increases its estimates, the top of the bar moves up.

- If additional items are added to the backlog, the bottom moves down; if items are taken out of the backlog or are replaced with lower-effort ones, the bottom moves up.

- For the same data as shown in above example, the Burndown Bar might look like this:

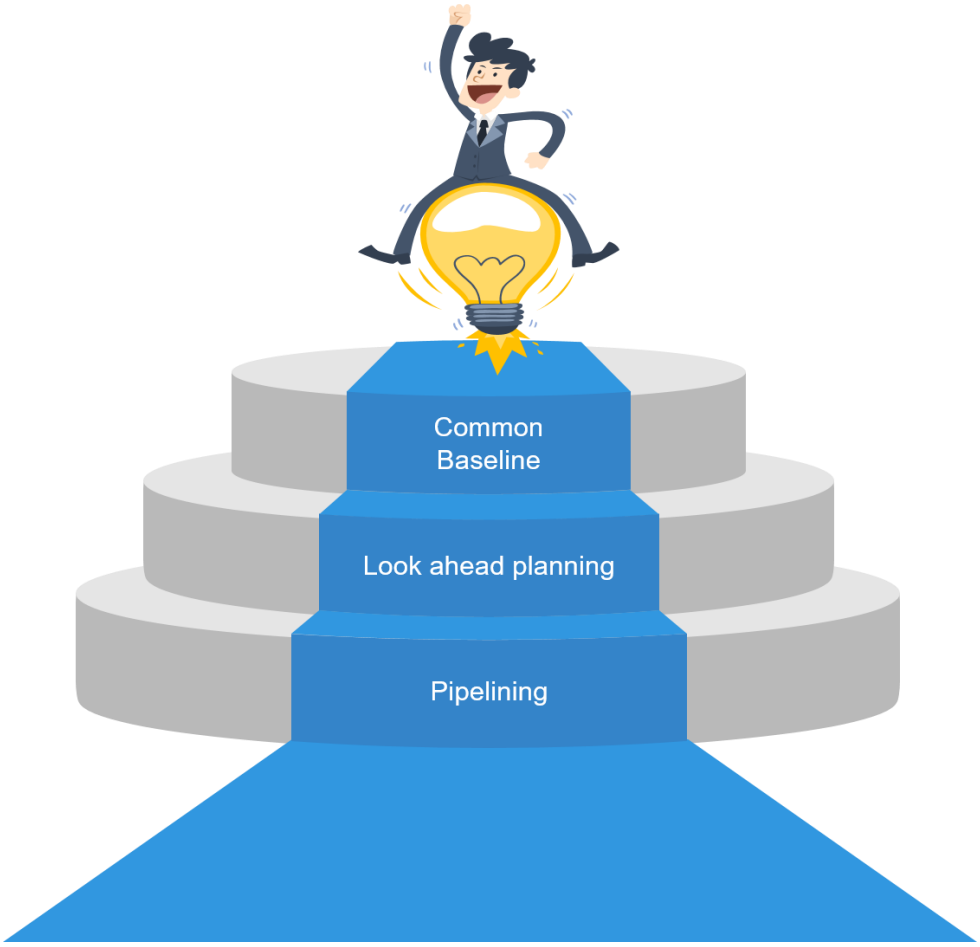


Release Plan

- Release Plan is like a map that guides you toward your destination.
- It helps to forecast when the Product is likely to become live and when its increments will be released.
- It provides more information than burndowns.
- The plan is based on four factors: the product backlog items, the remaining effort in the backlog, the velocity, and time.
- Example Release Plan is shown below:

Sprint →	1	2	3	4	5	6
Planned Velocity	20	25-30	25-30	30-35	30-35	30-35
Actual Velocity	18	22	…	…	…	…
Dependencies		Interface to Material Master				Reports and BI Dashboard Export
Releases			North America Release			EMEA Release

Large Project Release Planning



COMMON BASELINE

- On large projects, there is a possibility that multiple teams may be working on the same Product Backlog.
- In this situation, if each team comes up with their own ranges for Story Points, understanding the effort at Product Backlog level would be very difficult.
- In such cases, it is highly recommended that all teams can hold a joint effort estimation workshop, arrive and agree at a common range for Story Points with common semantics.
- In situations where a new team joins the project while the project is underway, the existing team can share their Story Point range and rationale behind that range. Both these teams can now come to a common understanding and use that Story Point range.

LOOKAHEAD PLANNING

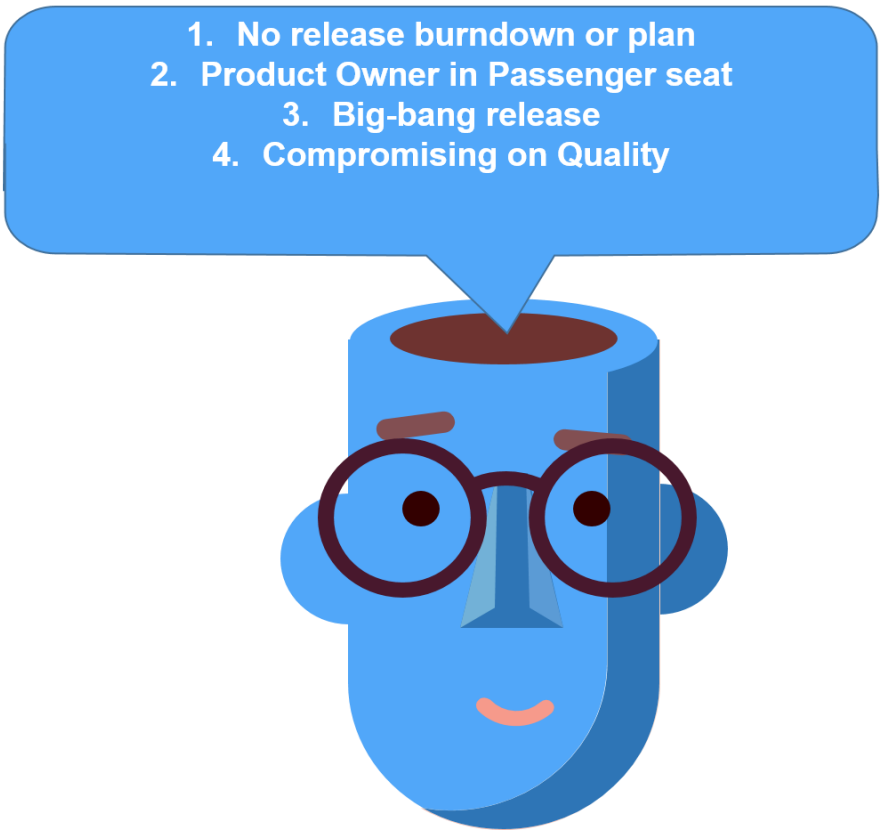
- As stated earlier, on a large project, the planning horizon can extend to next two to three Sprints.
- So, on a large project, Product Backlog items required for next two to three Sprints must be looked ahead—that is, planned early.

- After identifying these Backlog items, dependencies must be worked out. Since several teams are working together, dependencies must be sorted out as part of planning. This helps the teams plan for their part of work, the required interfaces if any, data exchange formats, and file formats for file transfer.
- Another factor to consider in lookahead planning is resource utilization. When there are several teams with dependencies between them on a large project, it is possible that a few teams are underutilized, while a few are overutilized. Therefore, in lookahead planning, resource utilization across teams also must be considered when planning a release.

PIPELINING

- This is the last option to try.
- Using pipelining, you can spread delivery of one Product Backlog item across multiple Sprints.
- For example, one team can work on a Product Backlog item in this Sprint. The remaining work of that Product Backlog item will be carried forward by another team in future Sprints.
- In pipelining, Feeding Buffers are used to create contingency between dependent teams.

Common Mistakes



NO RELEASE BURNDOWN OR PLAN

- A few organizations with their earlier practices of Waterfall project management, spend a lot of time in preparing a detailed project plan out of old habits and practices. With this project plan, they think Release Burndown is not necessary at all.
- They may also limit their planning to only one Sprint at a time on the other extreme.
- Such practices make it difficult to evaluate and assess project progress.
- To combat this common mistake:
    - Always have a release plan or release burndown
    - Make it visible to everyone either by displaying in team room or electronically

PRODUCT OWNER IN THE PASSENGER SEAT

- In a typical customer-supplier engagement, there is a possibility that Product Owner from the customer side may rely on Business Analysts or Project Manager from supplier side to come up with Product Backlog and Release Planning.
- In such situations, Product Owner delegates the ownership of Product Backlog to Supplier roles.
- To combat these common mistakes:
    - Ensure that Product Owner (from customer side) is engaged in the project full time
    - Product Owner owns the Product Backlog and drives release planning

BIG-BANG RELEASE

- A big-bang release is deploying or shipping the product ***only*** after it is completely developed; that is, when ***all*** the required functionalities are built.
- This makes it next to impossible to incorporate customer feedback, opportunities to inspect and adapt.
- To combat this common mistake:
    - Ensure that releases are frequent and early.

COMPROMISING ON QUALITY

- A few Product Owners, under deadline and launch date pressures, might be tempted to release more functionalities compromising on quality of the product.
- In such situations, progress might appear faster with limited testing. However, later a lot of rework would be required affecting the Product and the Brand image of the company.
- To combat this common mistake:
    - Ensure Definition of Done is clearly defined by Product Owner and agreed by the team.
    - During Sprint Review, Definition of Done is fully reviewed and tested upon. None of the acceptance criteria should be left untested.