Sprint Execution(II)

Dr. Elham Mahmoudzadeh

Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2023

Process

- The **inputs** to sprint execution are the sprint goal and the sprint backlog that were generated during sprint planning.
- The **output** from sprint execution is a potentially shippable product increment, which is a set of product backlog items completed to a high degree of confidence—where each item meets the Scrum team's agreed upon definition of done.
- **Activities** involves planning, managing, performing, and communicating the work necessary to create these working, tested features.

Which Work to Start

- Assuming that not all product backlog items are started simultaneously, at some point the team needs to determine which product backlog item to work on next.
- The simplest way to select the next product backlog item is to choose the next highest-priority item as specified by the product owner (via the item's position in the product backlog).
- This approach has the obvious advantage of ensuring that any items not completed during the sprint must be of lower priority than the ones that are completed.
- Unfortunately, the simplest approach won't always work because technical dependencies or skills capacity constraints might dictate that items be selected in a different order.
- The development team needs the ability to opportunistically make this selection as it sees fit.

How to Organize Task Work

- Once the development team decides to start working on a product backlog item, it must determine how to perform the task-level work on that item.
- If we apply waterfall thinking at the level of a single product backlog item, we would analyze the item, design it, code it, and then test it.
- Believing there is a single, predetermined, logical ordering to the work (for example, you have to build it before you can do any testing) blinds the team to the opportunity to do things in a different and perhaps more efficient way.

How to Organize Task Work(Cnt'd)

- Traditional role-based thinking plagues many teams. What we need instead is value-delivery-focused thinking, where the team members opportunistically organize the tasks and who will work on them.
- In doing so, they minimize the amount of time that work sits idle and reduce the size and frequency at which team members must "hand off" work to one another.
- This might mean, for example, that two people pair up on the first day of sprint execution and work in a highly interleaved fashion, with rapid cycles of test creation, code creation, test execution, and test and code refinement, and then repeat this cycle.
- This approach keeps work flowing (no blocked work), supports very fast feedback so that issues are identified and resolved quickly, and enables team members with T-shaped skills to swarm on an item to get it done.

What Work Needs to Be Done?

- What task-level work does the team perform to complete a product backlog item? Ultimately the team decides.
- Product owners and managers must trust that the team members are responsible professionals who have a vested interest in doing great work.
- As such, they need to empower these individuals to do the necessary work to create innovative solutions in an economically sensible way.
- Of course, product owners and managers do have influential input to what task level work gets done.
- First, the product owner ensures that the scope of a feature and its acceptance criteria are defined, both of which provide boundaries for the task-level work. Product owners and managers also provide business-facing requirements for the definition of done.
- Overall, the product owner must work with the team to ensure that technical decisions with important business consequences are made in an economically sensible way. Some of these decisions are embedded in the more technically oriented aspects of the definition of done.

What Work Needs to Be Done(Cnt'd)?

- Other decisions are feature specific. There is often a degree of flexibility regarding how much effort a team should exert on a feature.
- For example, enhancing or polishing a feature might be technically appealing but simply not worth the extra cost to the product owner at this time or ever.
- Conversely, cutting corners on a design or shortchanging where, how, or when we do testing also has economic consequences that must be considered.
- The team is expected to work with the product owner to discuss these trade-offs and make economically sensible choices.

Who Does the Work?

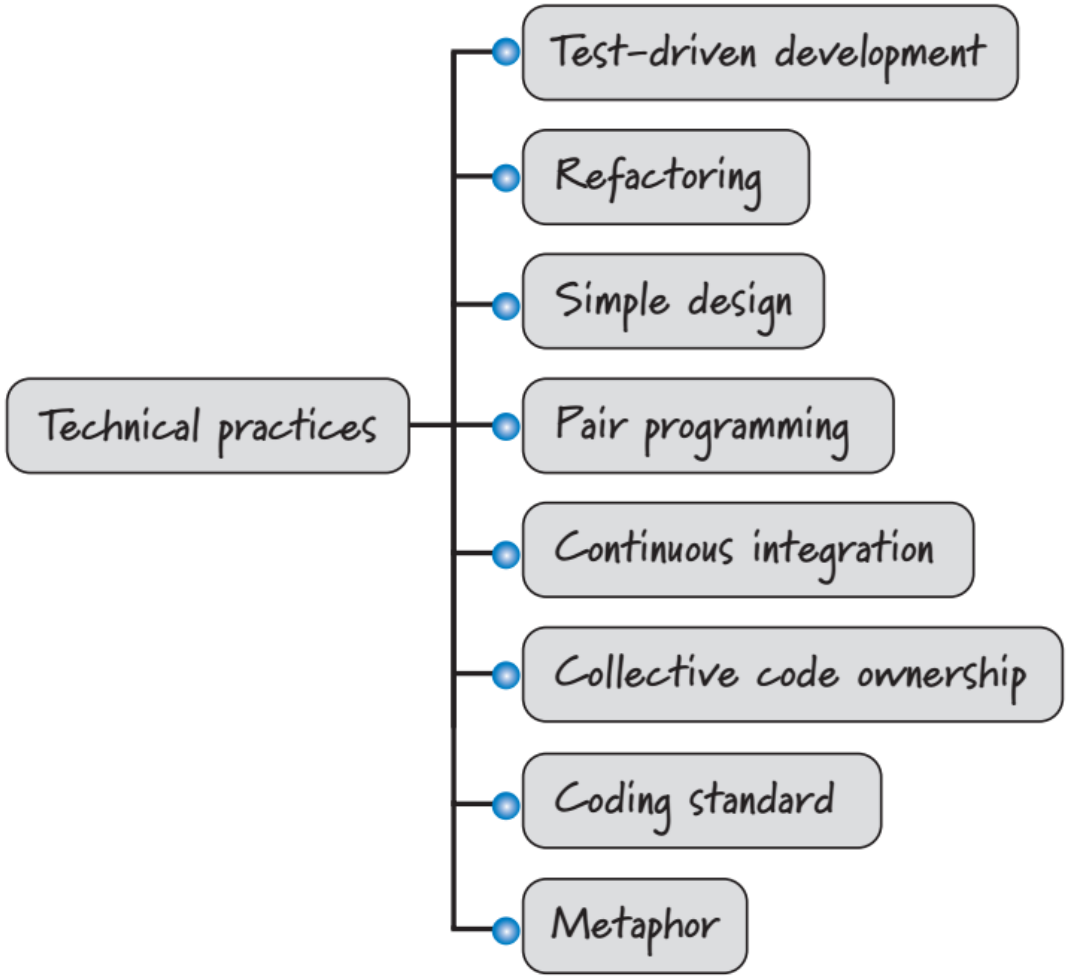
- Who should work on each task? An obvious answer is the person best able to quickly and correctly get it done. What if that person is unavailable? Perhaps she is already working on another, more important task, or maybe she is out sick and the task needs to get done immediately.
- There are a number of factors that can and should influence who will work on a task; it's the collective responsibility of the team members to consider those factors and make a good choice.
- When team members have T-shaped skills, several people on the team have the ability to work on each task. When some skills overlap among team members, the team can swarm people to the tasks that are inhibiting the flow of a product backlog item through sprint execution, making the team more efficient.

Daily Scrum

- Is a daily inspect-and-adapt activity to help the team achieve faster, more flexible flow toward the solution.
- Is a 15-minute, time-boxed activity that takes place once every 24 hours.
- Serves as an inspection, synchronization, and daily adaptive planning activity that helps a self-organizing team do its job better.
- The goal is for people who are focused on meeting the sprint goal to get together and share the big picture of what is happening so that they can collectively understand how much to work on, which items to start working on, and how to best organize the work among the team members.
- Helps avoid waiting. If there is an issue that is blocking flow, the team

Task Performance—Technical Practices

- Development team members are expected to be technically good at what they do.
- Working in short, time-boxed iterations where there is an expectation of delivering potentially shippable product increments does exert pressure on teams to get the job done.
- If team members lack appropriate technical skills, they will likely fail to achieve the level of agility needed to deliver long-term, sustainable business value.
- If you are using Scrum to develop software, team members need to be skilled in good technical practices for developing software.


Technical practices
Test-driven development
Refactoring
Simple design
Pair programming
Continuous integration
Collective code ownership
Coding standard
Metaphor

Process

- The **inputs** to sprint execution are the sprint goal and the sprint backlog that were generated during sprint planning.
- The **output** from sprint execution is a potentially shippable product increment, which is a set of product backlog items completed to a high degree of confidence—where each item meets the Scrum team's agreed upon definition of done.
- **Activities** involves planning, managing, performing, and communicating the work necessary to create these working, tested features.

Communicating

- One of the benefits of working in short time-boxes with small teams is that you don’t need complex charts and reports to communicate progress!
- Although any highly visible way of communicating progress can be used, most teams use a combination of a task board and a burndown and/or burnup chart as their principal information radiator.

Task Board

- The task board is a simple but powerful way to communicate sprint progress at a glance.
- Formally, the task board shows the evolving state of the sprint backlog over time.

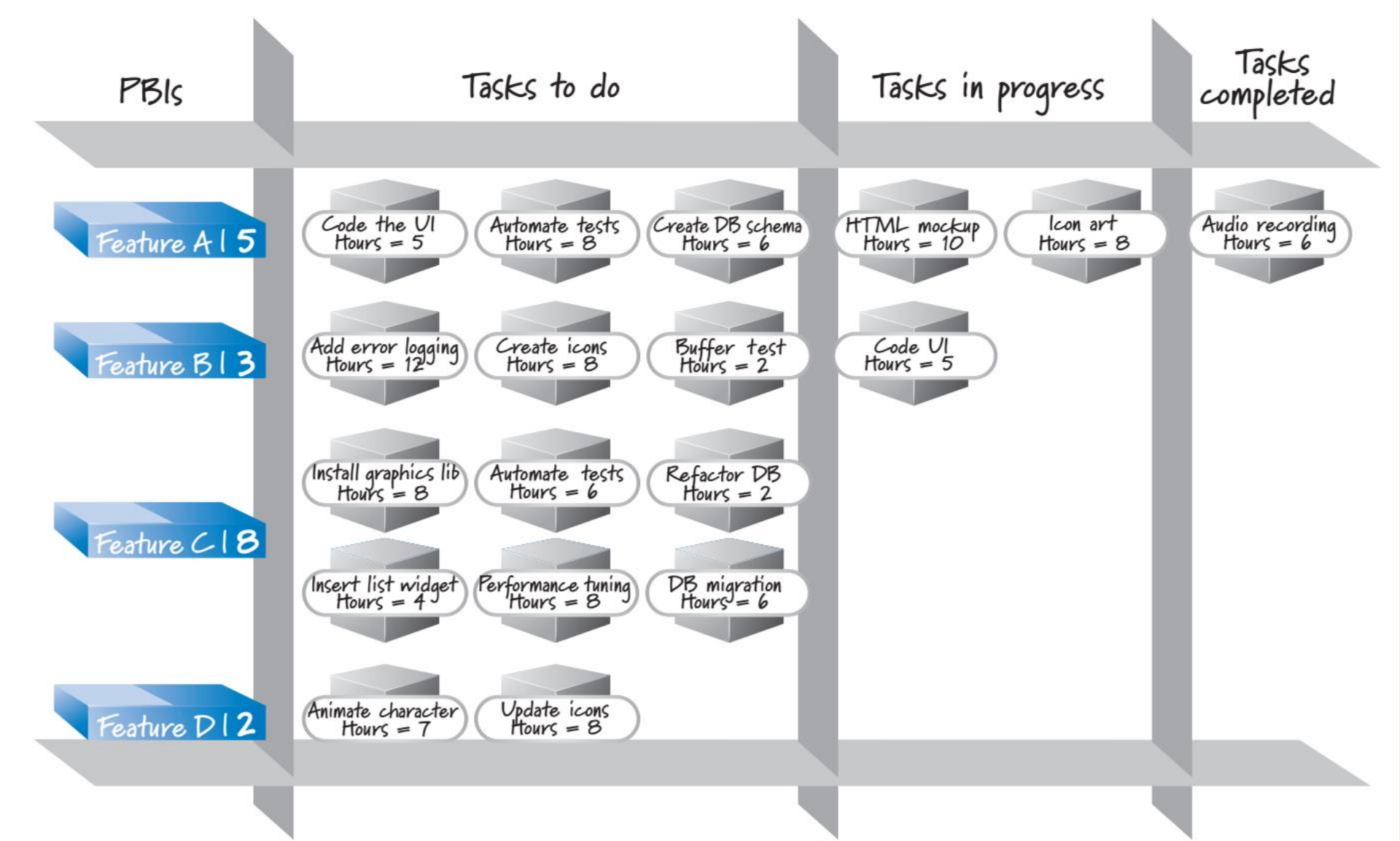


FIGURE 20.6 Example task board

Task Board

- On this task board each product backlog item planned to be worked on during the sprint is shown with the set of tasks necessary to get the item done.
- All tasks initially start off in the “to do” column.
- Once the team determines that it is appropriate to work on an item, team members start selecting tasks in the “to do” column for the item and move them into the “in progress” column to indicate that work on those tasks has begun.
- When a task is completed, it is moved to the “completed” column.

Sprint Burndown Chart

- Each day during sprint execution team members update the estimate of how much effort remains for each uncompleted task.
- We could create a table to visualize this data.
- Number of hours remaining for each task follows the general trend of being smaller each day during the sprint—because tasks are being worked on and completed.
- If a task hasn't yet been started (it is still in the task board "to do" column), the size of the task might appear the same from day to day until the task is started.
- Of course, a task might turn out to be larger than expected, and if so, its size may actually increase day over day or remain the same size even after the team has started working it.

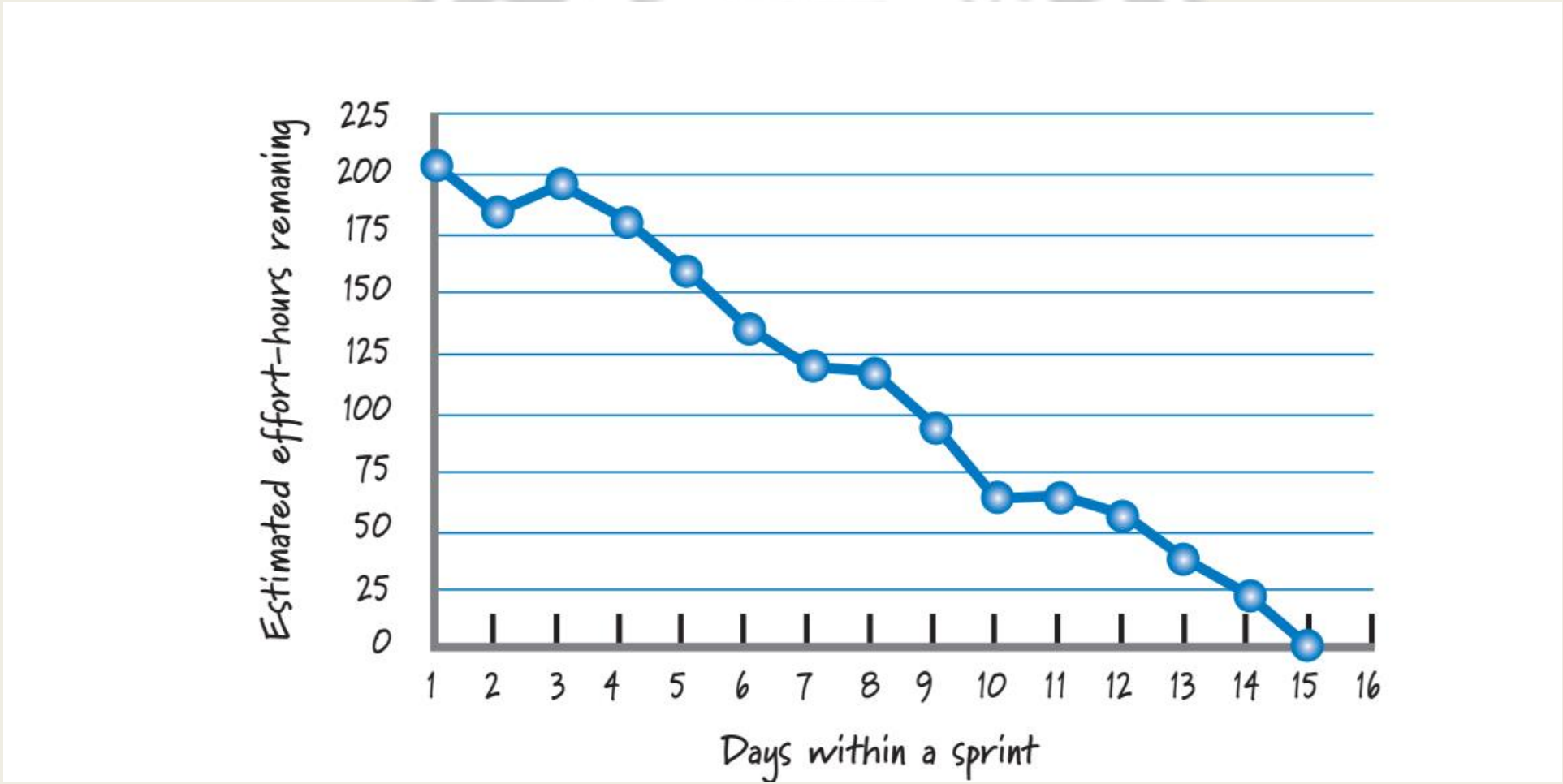
TABLE 20.1 Sprint Backlog with Estimated Effort Remaining Each Day

Tasks	D1	D2	D3	D4	D5	D6	D7	D8	D9	. . .	D15
Task 1	8	4	4	2							
Task 2	12	8	16	14	9	6	2				
Task 3	5	5	3	3	1						
Task 4	7	7	7	5	10	6	3	1			
Task 5	3	3	3	3	3	3	3				
Task 6	14	14	14	14	14	14	14	8	4		
Task 7						8	6	4	2		
Tasks 8–30	151	139	143	134	118	99	89	101	84		0
Total	200	180	190	175	155	130	115	113	90		0

Sprint Burndown Chart(Cnt'd)

- New tasks related to the committed product backlog items can also be added to the sprint backlog at any time. There is no reason to avoid adding a task to the sprint backlog.
- It represents real work that the team must do to complete a product backlog item that the team agreed to get done.
- Permitting unforeseen tasks to be added to the sprint backlog is not a loophole for introducing new work into the sprint.
- It simply acknowledges that during sprint planning we may not be able to fully define the complete set of tasks needed to design, build, integrate, and test the committed product backlog items. As our understanding of the work improves by doing it, we can and should adjust the sprint backlog.

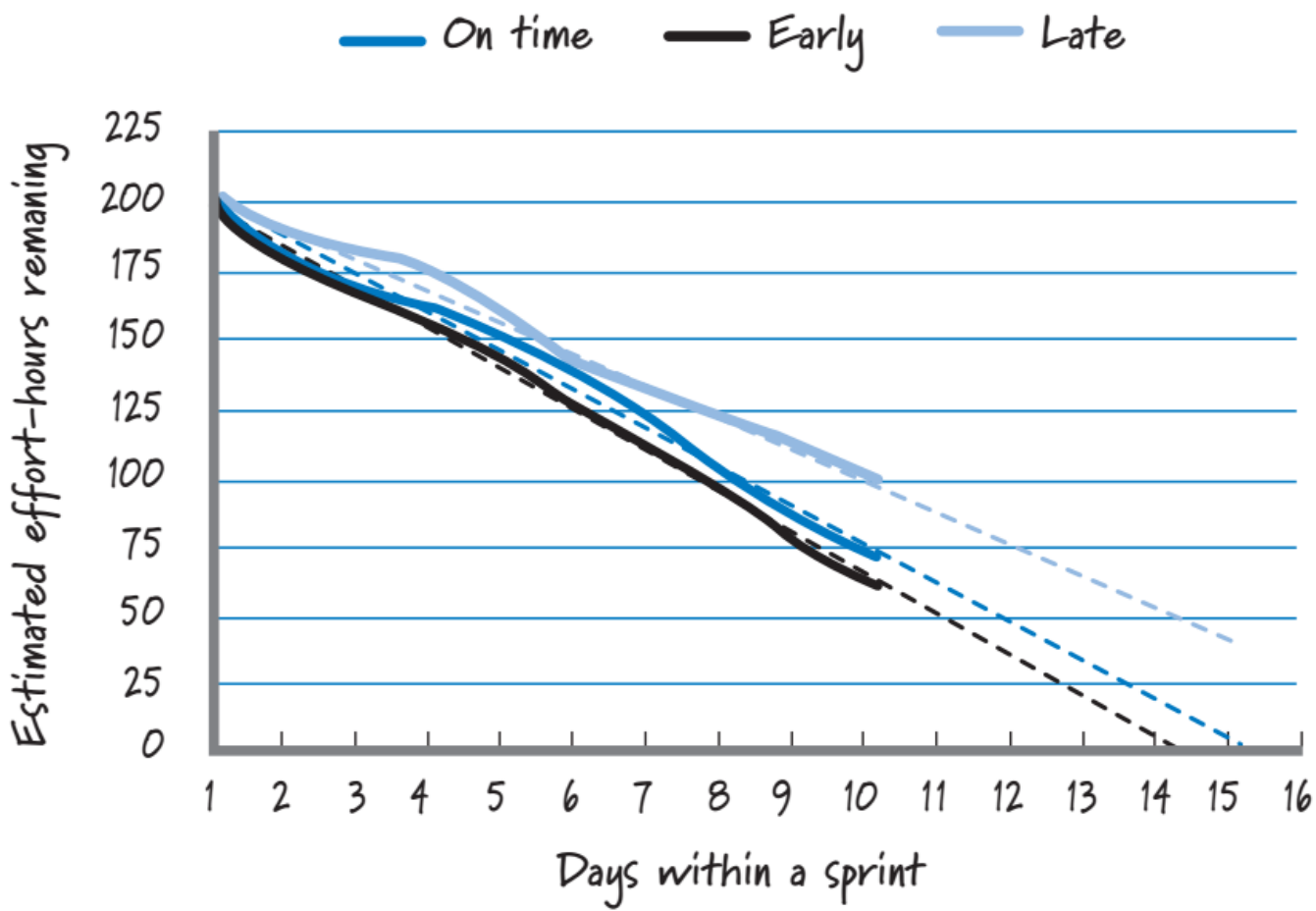
Sprint Burndown chart



Sprint burndown charts are useful for tracking progress and can also be used as a leading indicator to predict when work will be completed.

At any point in time we could compute a trend line based on historical data and use that trend line to see when we are likely to finish if the current pace and scope remain constant.

Sprint Burdown chart with trend lines



Sprint Burndown chart with trend lines

- When the trend line intersects the horizontal axis close to the end of the sprint duration, we can infer that we're in reasonable shape ("On time").
- When it lands significantly to the left, we should probably take a look to see if we can safely take on additional work ("Early").
- But when it lands significantly to the right ("Late"), that raises a flag that we're not proceeding at the expected pace or that we've taken on too much work (or both!). When that happens, we should dig deeper to see what's behind the data and what, if anything, needs to be done.

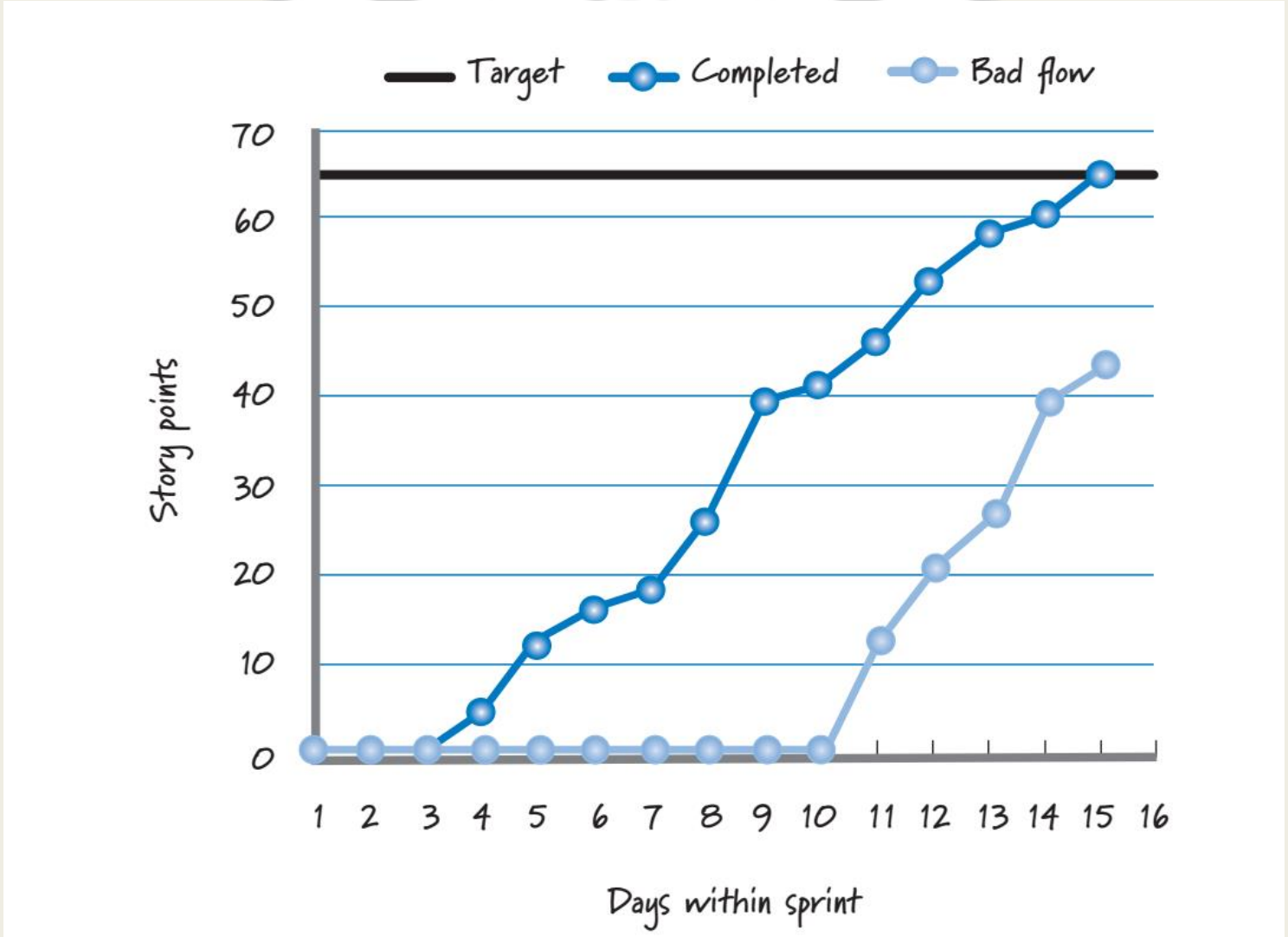
Sprint Burndown chart(Cnt'd)

- By projecting the trend lines, we have another important set of data that adds to our knowledge of how we are managing flow within our sprint.
- The sprint backlog and the sprint burndown charts always use estimated effort *remaining*.
- They do not capture actual effort expended. In Scrum there is no specific need to capture the *actuals*; however, your organization might choose to do so for non-Scrum reasons such as cost accounting or tax purposes.

Sprint Burnup Chart

- Is an alternative way to visualize progress through a sprint.
- Represent the amount of work completed toward achieving sprint goal

Sprint Burnup chart(Cnt'd)



Sprint Burnup chart(Cnt'd)

- In sprint burnup charts the work can be represented in either effort-hours (as in the sprint burndown chart) or in story points.
- Many people prefer to use story points in their burnup charts, because at the end of the sprint the only thing that really matters to the Scrum team is business-valuable work that was completed during the sprint, and that is measured in story points (or ideal days), not task hours. Also, if we measure story points of completed product backlog items, at a glance we can get a good feel for how the work is flowing and how the team is completing product backlog items through the sprint.
- To illustrate this point a third line (labeled "Bad flow") is included on the sprint burnup chart. The "Bad flow" line illustrates what the burnup chart might look like if the team starts too many product backlog items at the same time, delays completion of items until later in the sprint, fails to meet the sprint target because of the reduced velocity of doing too much work in parallel, works on product backlog items that are large and therefore take a long time to finish, or takes other actions that result in bad flow.

Reference

1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.