



Everyone's Smart
University

PMG 622

AGILE PROJECT MANAGEMENT

LECTURE 05: IMPLEMENTING AGILE: DELIVERING IN AN AGILE ENVIRONMENT **[PART 2 OF 2]**

دكتور مهندس استشاري ادارة المشروعات

Eng. Dr. Akram Mustafa Suliman Ali,

PhD. AU, BEng. & MSc. AAST, CEng CMarEng MIMarEST, PE, PMP®,

ASQ CSSBB, ISO Lead Auditor, NDT Level II,



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

QUICK REVIEW AND INTRODUCTION

SCRUM ROLES, ARTIFACTS & CEREMONIES



- SCRUM ARTIFICATS
 - Product Backlog
 - Sprint Backlog
 - Increment
- SCRUM CERMONIES
 - Sprint Planning.
 - Daily **Scrum**.
 - Sprint Review.
 - Sprint Retrospective.
- SCRUM ROLES
 - Product Owner
 - Development Team Members
 - Scrum Master


Product Owner


Scrum Master


Scrum Team Members


Stakeholders


Project Vision


Epics


User Stories


Release Plan


Sprint Review Meeting


Sprint Retrospective Meeting


Planning
Sprint Planning Meeting


Daily Scrum


Impediment Log


Burndown Chart


Project Retrospective Meeting

SPRINT

Visual  Paradigm

DAILY STAND-UP QUESTIONS

iteration-based agile (TIME-BOXED) SCRUM	flow-based agile (INCREMENT) KANBAN
• What did I complete since the last standup?	• What do we need to do to advance this piece of work?
• What am I planning to complete between now and the next standup?	• Is anyone working on anything that is not on the board?
• What are my impediments (or risks or problems)?	• What do we need to finish as a team?
	• Are there any bottlenecks or blockers to the flow of work?



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

TROUBLESHOOTING AGILE PROJECT CHALLENGES

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [1/8]

Pain Point	Troubleshooting Possibilities
Unclear purpose or mission for the team	Agile chartering for purpose–vision, mission, and mission tests
Unclear working agreements for the team	Agile chartering for alignment–values, principles, and working agreements
Unclear team context	Agile chartering for context–boundaries, committed assets, and prospective analysis

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [2/8]

Pain Point	Troubleshooting Possibilities
Unclear requirements	❖ Help sponsors and stakeholders craft a product vision. ❖ Consider building a product roadmap using specification by example, user story mapping, and impact mapping. ❖ Bring the team and product owner together to clarify the expectations and value of a requirement. ❖ Progressively decompose roadmap into backlog of smaller, concrete requirements.
Poor user experience	User experience design practices included in the development team involve users early and often.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [3/8]

Pain Point	Troubleshooting Possibilities
Inaccurate estimation	❖ Reduce story size by splitting stories. ❖ Use relative estimation with the entire team to estimate. ❖ Consider agile modeling or spiking to understand what the story is.
Unclear work assignments or work progress	❖ Help the team learn that they self-manage their work. ❖ Consider kanban boards to see the flow of work. ❖ Consider a daily standup to walk the board and see what work is where.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [4/8]

Pain Point	Troubleshooting Possibilities
Team struggles with obstacles	❖ A servant leader can help clear these obstacles. ❖ If the team doesn't know the options they have, consider a coach. ❖ Sometimes, the team needs to escalate stories the team or servant leader has not been able to remove.
Work delays/overruns due to Insufficiently refined product backlog items	❖ Product owner and team workshop stories together. ❖ Create a definition of ready for the stories. ❖ Consider splitting stories to use smaller stories
Defects	❖ Consider the technical practices that work for the environment. ❖ Some possibilities are: pair work, collective product ownership, pervasive testing (test-driven and automated testing approaches) and a robust definition of done.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [5/8]

Pain Point	Troubleshooting Possibilities
Work is not complete	❖ Team defines definition of done for stories including acceptance criteria. ❖ Also add release criteria for projects.
Technical debt (degraded code quality)	Refactoring, agile modeling, pervasive testing, automated code quality analysis, definition of Done
Slow or no improvement in the teamwork process	❖ Capture no more than three items to improve at each retrospective. ❖ Ask the servant leader to help the team learn how to integrate those items.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [6/8]

Pain Point	Troubleshooting Possibilities
Too much product complexity	For software and non-software encourage the team always to be thinking “What is the simplest thing that would work?” and apply the agile principle of <u>“Simplicity--the art of maximizing the amount of work not done”</u> . These help reduce complexity.
Too much upfront work leading to rework	❖ Instead of much upfront work, consider team spikes to learn. ❖ In addition, measure the WIP during the beginning of the project and see what the team's options are to deliver value instead of designs. ❖ Shorten iterations and create a robust definition of done.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [7/8]

Pain Point	Troubleshooting Possibilities
False starts, wasted efforts	Ask the product owner to become an integral part of the team.
Inefficiently ordered product backlog items	Rank with value including cost of delay divided by duration and other value models
Rush/wait uneven flow of work	❖ Plan to the team's capacity and not more. ❖ Ask people to stop multitasking and be dedicated to one team. ❖ Ask the team to work as pairs, a swarm, or mob to even out the capabilities across the entire team.

AGILE PAIN POINTS AND TROUBLESHOOTING POSSIBILITIES [8/8]

Pain Point	Troubleshooting Possibilities
Impossible stakeholder demands	Servant leadership to work with this stakeholder (and possibly product owner).
Unexpected or unforeseen delays	❖ Ask the team to check in more often, use Kanban boards to see the flow of work and work in progress limits to understand the impact of the demands on the team or product. ❖ Also track impediments and impediment removal on an impediment board.
Siloed teams, instead of cross-functional teams	❖ Ask the people who are part of projects to selforganize as cross-functional teams. ❖ Use servant leadership skills to help the managers understand why agile needs cross-functional teams.



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

MEASUREMENTS IN AGILE PROJECTS

TRANSITIONING TO AGILE

- Using different measurements
- Looking at new metrics that matter to the team and to management
- These metrics matter because they focus on customer value

AGILE TEAMS MEASURE RESULTS

- Agile favors empirical and value-based measurements instead of predictive measurements.
- Agile measures what the team delivers, not what the team predicts it will deliver.
- Agile is based on working products of demonstrable value to customers and not managing to a baseline

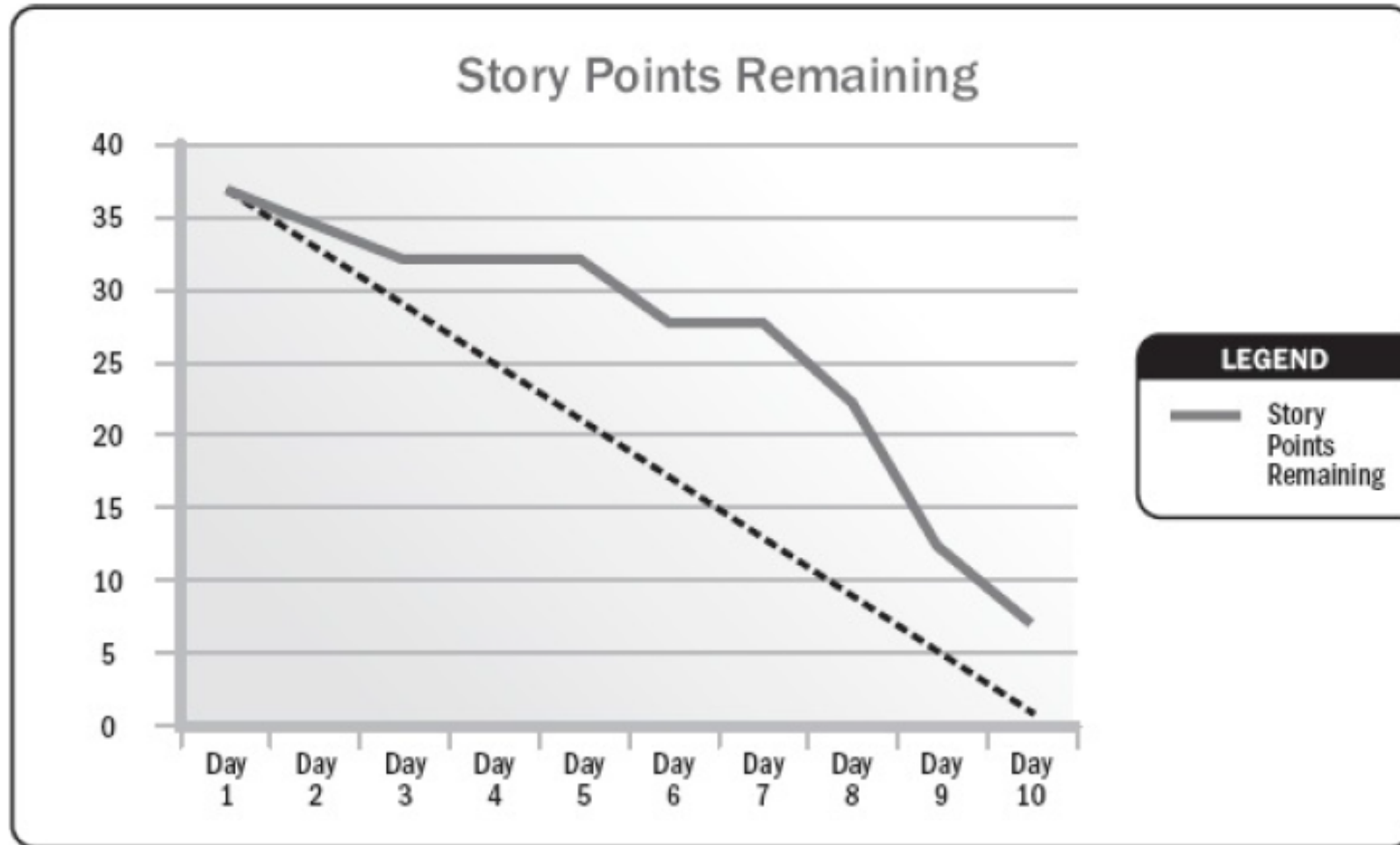
AGILE DURATION PREDICTION

- Once the team establishes a reliable velocity (average stories or story points per iteration) or the average cycle time, the team can predict how much longer the project will take
- Teams might discover it can take four to eight iterations to achieve a stable velocity. The teams need the feedback from each iteration to learn about how they work and how to improve.

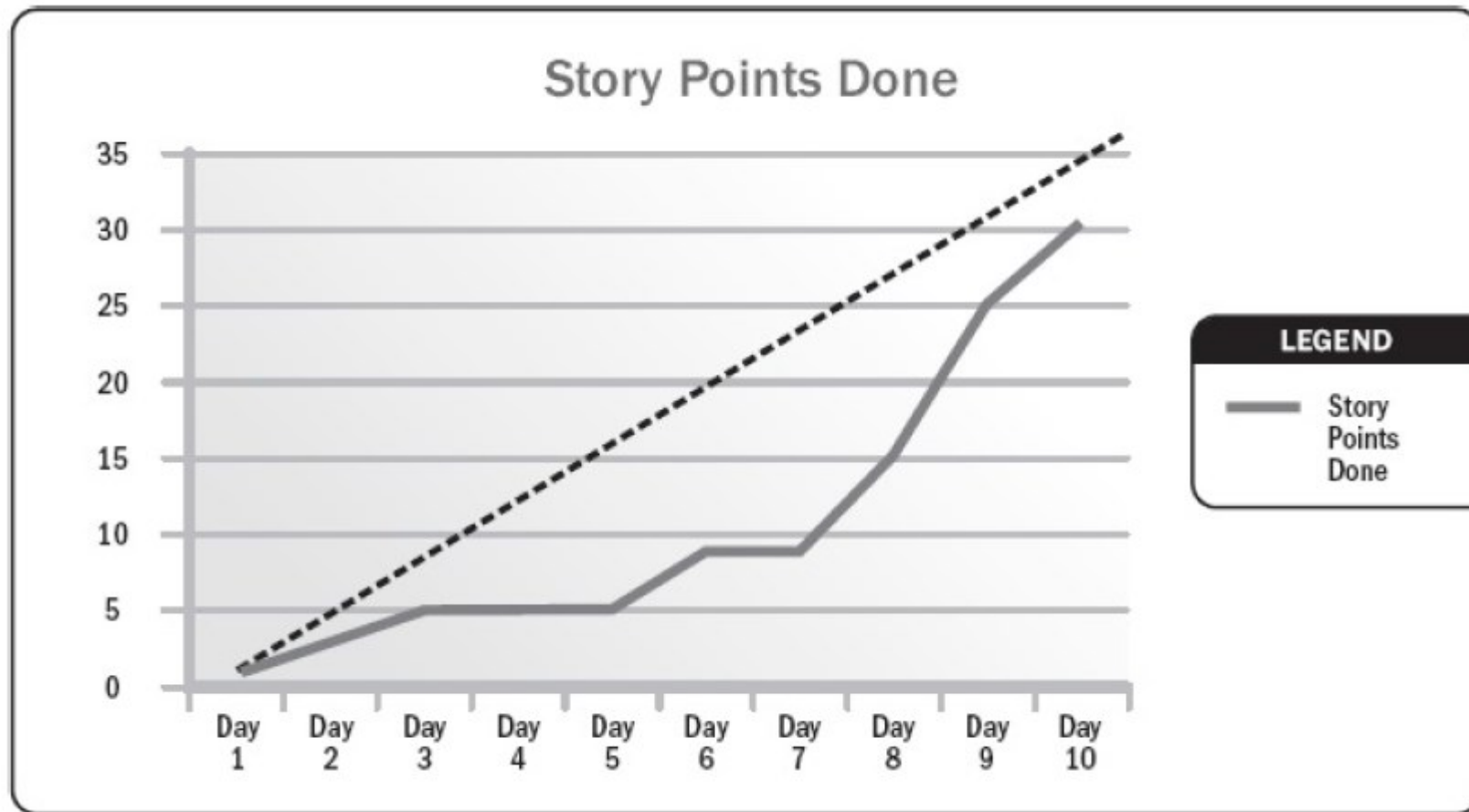
VELOCITY,

- The sum of the story point sizes for the features actually completed in this iteration, allows the team to plan its next capacity more accurately by looking at its historical performance.

SPRINT BURNDOWN CHARTS



SPRINT BURNUP CHARTS



BURNDOWN AND BURNUP CHARTS

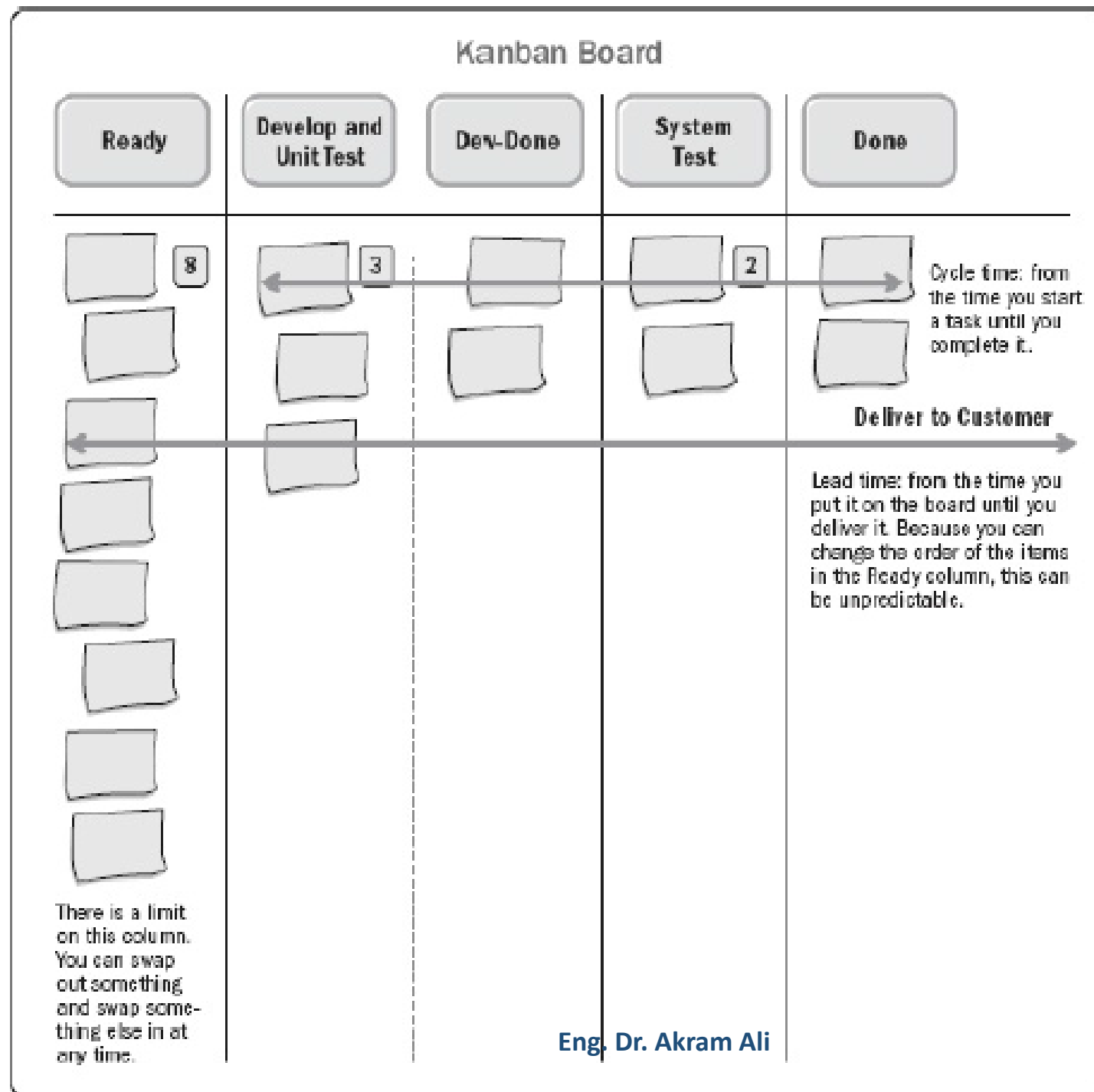
- Burndowns show the effect of team members multitasking, stories that are too large, or team members out of the office.
- Especially with teams new to agile, the burnup will show changes in scope during the iteration. Burnups allow teams to see what they have accomplished, which helps the team proceed to the next piece of work.

FLOW-BASED AGILE TEAMS MEASUREMENTS:

- Lead time (the total time it takes to deliver an item, measured from the time it is added to the board to the moment it is completed)
- Cycle time (the time required to process an item)
- Response time (the time that an item waits until work starts)
- Teams measure cycle time to see bottlenecks and delays, not necessarily inside the team.

TIP

- *When teams depend on external people or groups, measure cycle time to see how long it takes for the team to complete the work.*
- *Measure lead time to see the external dependencies after the team completes its work.*
- *Teams can also measure the reaction time, the time from ready to the first column, to see how long it takes them—on average—to respond to new requests.*



When the team has met its WIP limits, the team cannot pull work from the left into the next column. Instead, the team works from the right-most full column and asks, “What do we do as a team to move this work into the next column?”

CAPACITY MEASURES & PREDICTABILITY MEASURES

- Burnups, burndowns (capacity measures) and lead time, and cycle time (predictability measures) are useful for in-the-moment measurements. They help a team understand how much more work they have and whether the team might finish on time.
- Measuring story points is not the same as measuring completed stories or features. Some teams attempt to measure story points without completing the actual feature or story. When teams measure only story points, they measure capacity, not finished work, which violates the principle of “the primary measure of progress is working software” (or, other product if not software).
- Each team has its own capacity. When a team uses story points, be aware that the number of story points a team can complete in a given time is unique to that team.
- When teams provide their own units of measure, teams are better able to assess and estimate and deliver their work.

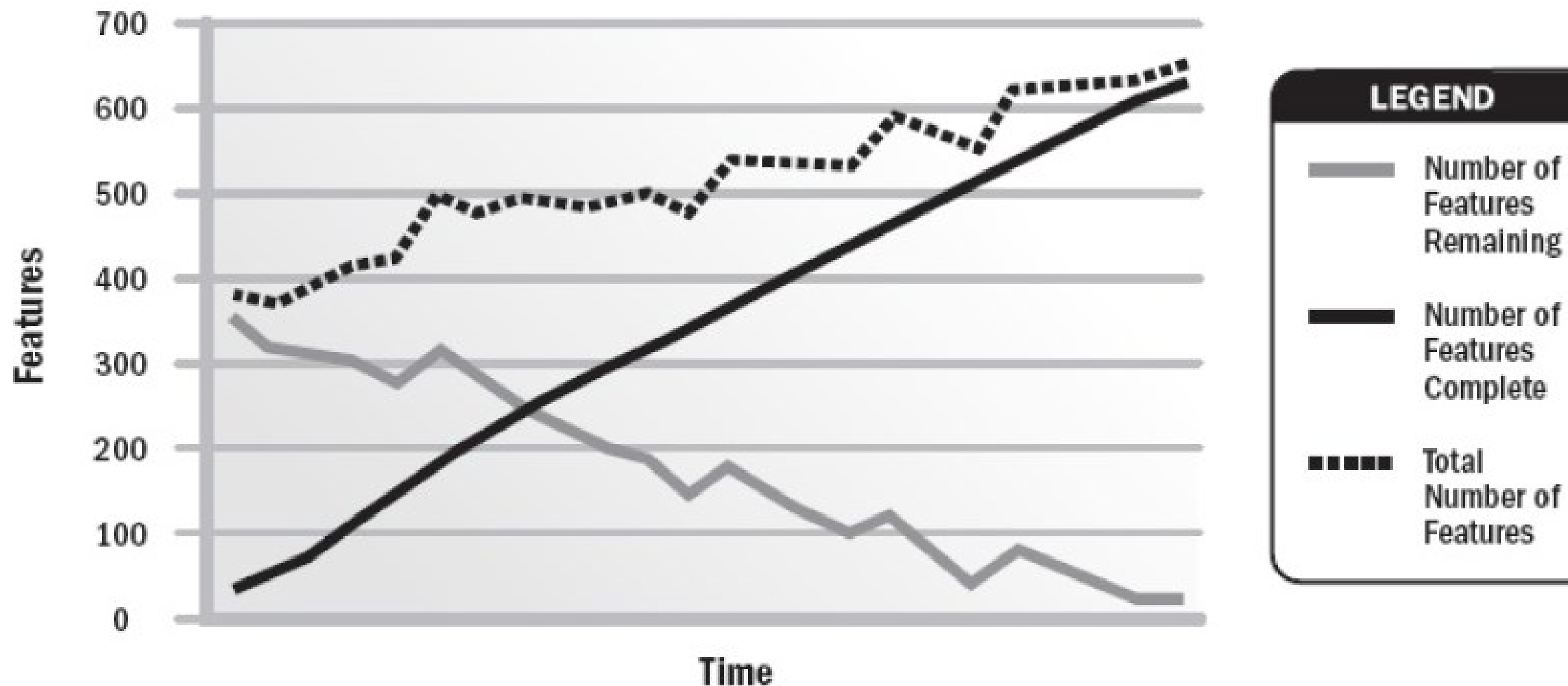
THE DOWNSIDE OF RELATIVE ESTIMATION

- There is no way to compare teams or add velocity across teams

FEATURE BURNUP/BURNDOWN CHART

- These charts provide trends of completion over time
- Show that requirements grew during the project.
- The features complete line shows that the team completes features at a regular pace.
- The total features line shows how the project's total features changed over time.
- The features remaining burndown line shows that the rate of feature completion varies.
- Every time features are added to the project, the burndown line changes.

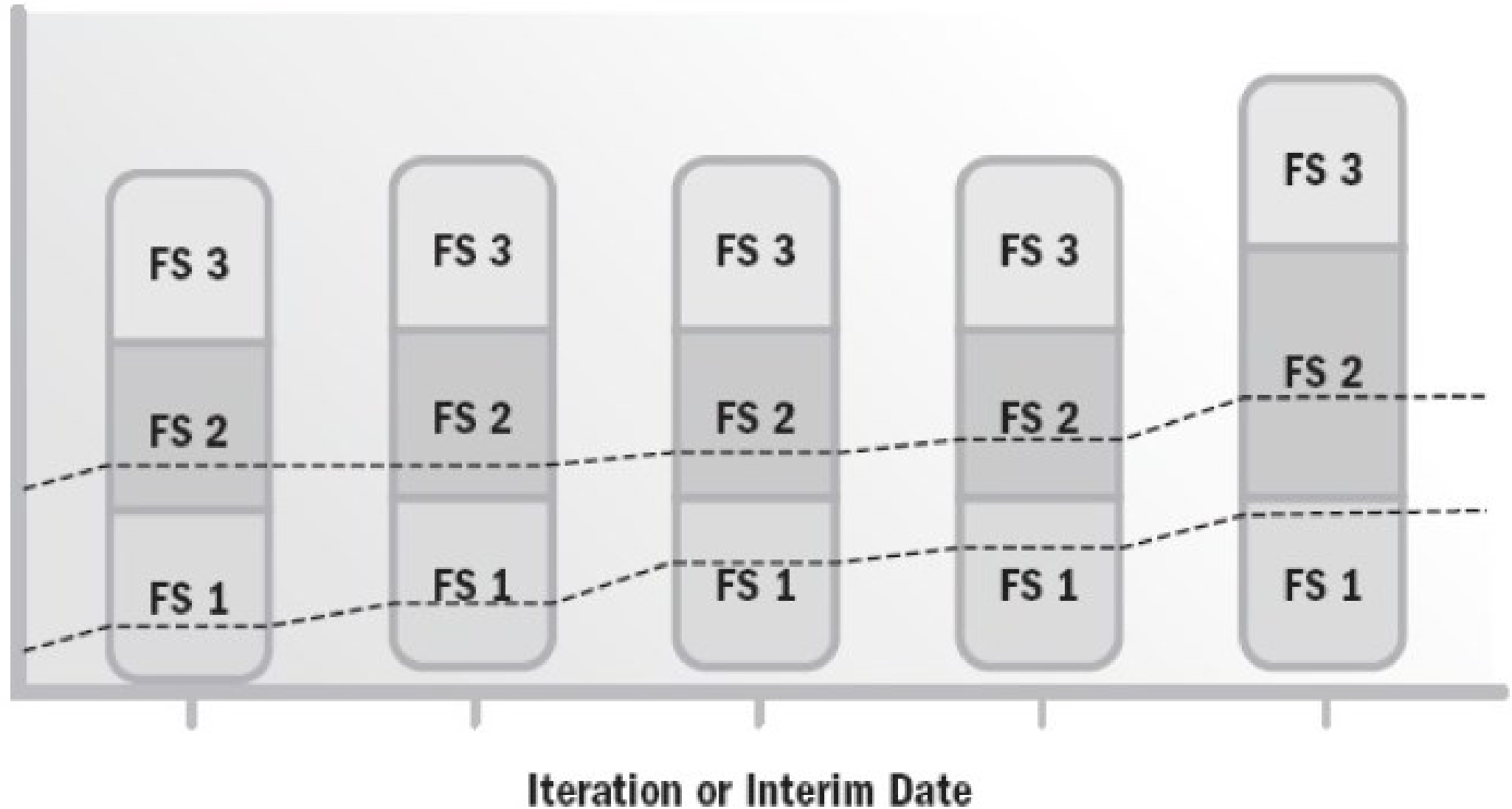
Features Complete, Remaining and Total



PRODUCT BACKLOG BURNUP CHART

- Earned value in agile is based on finished features
- The product backlog burnup chart shows completed work compared to total expected work at interval milestones or iterations.
- A team can only finish one story at a time.
- To complete a large feature that contains several stories, the team will have remaining stories to complete and may not complete that entire feature until several more time periods have passed.
- The team can show its completed value with a product backlog burnup chart

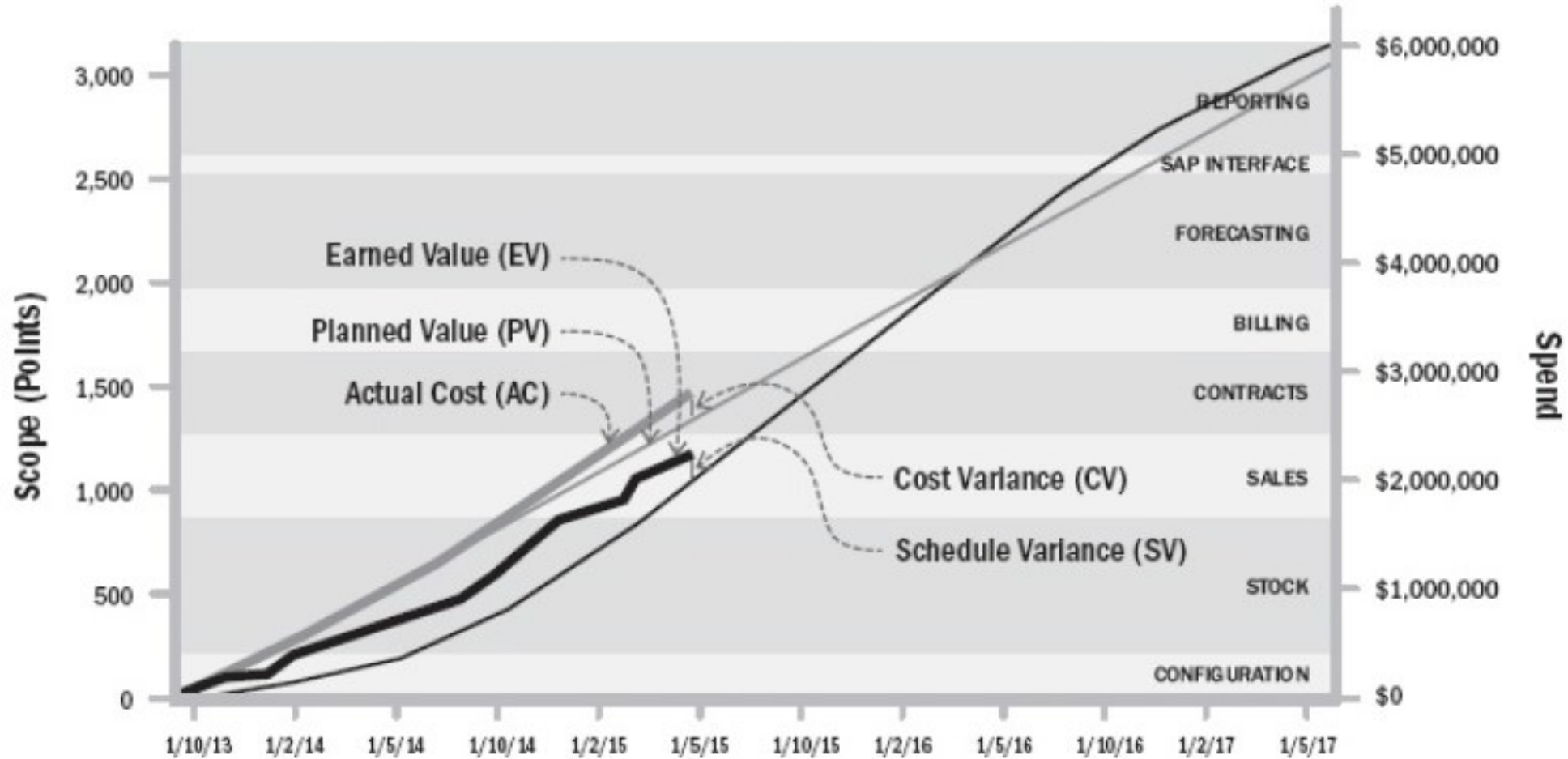
Cumulative Features



EARNED VALUE IN AN AGILE CONTEXT

- Traditional EVM metrics like schedule performance index (SPI) and cost performance index (CPI) can be easily translated into agile terms.
- For example, if the team planned to complete 30 story points in an iteration, but only completed 25 then the SPI is $25/30$ or 0.83 (the team is working at only 83% of the rate planned).
- Likewise, CPI is the earned value (completed features value) to date divided by the actual costs to date or, as shown in Figure, $\$2.2\text{M} /$
- $\$2.8\text{M} = 0.79$. This means a result of only 79 cents on the dollar compared to plan

ABC Project Progress



$$SPI = \frac{\text{Completed Features}}{\text{Planned Features}}$$

$$CPI = \frac{\text{Earned Value}}{\text{Actual Costs}}$$

LEGEND

Projected Progress Scope Built Projected Spend Spend