Iteration and Release Planning

CSCI 5828: Foundations of Software Engineering
Lecture 15 — 10/13/2015

Goals

- Estimating User Stories
- Planning a Release
- Planning an Iteration
- Measuring and Monitoring Velocity

Estimating User Stories

- Developers need to assign “points” to a story to indicate how long it will take to implement
    - Our user/customer assigns priorities to stories, not estimates
- There are a number of desirable properties for this approach
    - it allows us to change our minds about an estimate when new info arrives
    - works for both epic stories as well as smaller stories
    - doesn’t take a lot of time; we want to spend our time developing
    - provides useful information about our progress and work remaining
    - is tolerant of imprecision in estimates
    - can be used to plan releases

Story Points

- A point is a unit that can be defined by the development team
    - It might represent “eight hours of uninterrupted work” for one team
    - It might represent “forty hours of uninterrupted work” for another
    - Some use points to represent complexity (lots of points == complex)
- Think of one point as “one ideal work day”
    - where ideal means: a day with no interruptions and the developer can be maximally productive on the task
- Two benefits with this approach
    - it avoids getting too specific: “this story will take 39.5 hours”
    - it gives people confidence: “Yeah, that story is about two days of work”

Estimates belong to the Team

- It is important to have the team create the estimates for each story
    - The success of the project is attributed to the team not to individuals
        - to establish this perspective: make estimates together
            - if you get it wrong, it's the team that failed, not one individual
- In addition, when creating/estimating stories, it may not be clear who will be assigned to this particular story
    - therefore, the team works to create the estimate and then individuals assigned to the story later know
        - they had a voice in creating the estimate they are working against
        - the team is responsible if the estimate is wrong

The Process of Estimation

- One way to do estimation was developed by Barry Boehm
  - the Wideband Delphi approach
- Gather the development team and the customer/user(s)
  - Bring the stories that need estimates and blank index cards
  - Distribute the cards to the development team
- **Loop until all stories have estimates**
  - Read a story out-loud
  - **Loop until estimates have converged**
    - Engage in Q&A with customer/users about that story
    - Each developer writes an estimate; when ready, show all estimates
    - Developers discuss differences in estimates; raising questions/issues
      - New stories may be created due to this discussion

Triangulate

- After a set of stories have received estimates, developers need to review them and see if they are being consistent
    - Group the stories by number of points and discuss
        - For example, are these two point stories really twice as small as the four points stories?
            - If yes, continue estimating
            - If not, change the estimates
- This helps the team achieve consistency across the entire set of user stories
    - Later in a development project, the need for triangulation may go down as the team becomes more confident and knowledgable of their abilities

Velocity

- The term velocity is defined as “number of story points completed per iteration”
    - Agile software life cycles recommend that
        - before the first iteration begins, the team makes a guess at what their velocity will be
            - if a point means “ideal work day”, you can start with this formula
                - number of team members x number of days in iteration
        - then, your velocity for iteration N is the actual number of points completed for iteration N-1
            - if you completed 32 points in the previous iteration, your velocity for planning the next iteration is 32.

Release Planning

- A release is a version of the system under development that is going to be deployed and put into production use
  - Release planning in software development involves having a release roadmap in which the next several releases have been identified
    - and the functionality for each release has been specified at a high level
    - Kent Beck recommends thinking of this as “themes” for each release
- With a release roadmap, you need to engage in release planning
  - users/customers need to assign priorities to estimated user stories
  - all stakeholders need to work together to identify the length of an iteration
  - Issues include dealing with risk and determining velocity

Assigning Priorities

- One prioritization scheme that may be better than the typical "low/medium/high" approach
  - Must have
  - Should have
  - Could have
  - Won't have (for this release)
- This approach divides stories into clear buckets that can then be used to assign stories to iterations within the release
  - If a customer can't assign a priority to a user story, this (typically) indicates that the story needs to be split until clear priorities can be assigned

Risky Stories

- The issue here is what approach should agile projects take
    - tackle risky stories first
    - or go after “low hanging fruit”
- Agile life cycles like to go after low-hanging fruit
    - ***high-value functionality that is straightforward to implement***
- This allows time for more information to be gathered about high-risk stories
    - and this additional information may reduce the risk associated with them
- I think you need to balance this with the common issue of “problem avoidance”; make sure you’re clear on what the risks are => such information may produce action items that can reduce the risk and make it feasible

Iteration Length and Expected Duration

- Iteration length is typically from one week to four weeks
  - Agile life cycles recommend selecting shorter lengths to increase the feedback loop with the customer
- The important thing is once the length is selected: **DON'T CHANGE IT!**
  - Your team needs to settle into a comfortable development pace
    - Arbitrary changes to the iteration length will hinder that goal
- Once you have an iteration length, an initial velocity, and a set of prioritized, estimated user stories, you can make initial "ballpark" predictions about how long it will take to create a release
  - round_up(number of points / velocity) == number of iterations
  - number_of_iterations * iteration_length == number of days until release

Velocity, revisited

- Previously we suggested
  - number of team members x number of days in iteration
- is a good formula for picking an initial velocity
- However, you need to take into account that “number of days” means “number of **IDEAL** days”
  - You need to include a conversion factor between an IDEAL day and an ACTUAL day
    - An actual day won’t be eight hours of uninterrupted work due to meetings, interruptions, illness, turnover, etc.
- Ideal velocity for six people with two week iteration (10 business days): 60
- Converting to an ACTUAL day: 6 x 10 x .5 = 30; 6 x 10 x .25 = 15!

Iteration Planning (I)

- The points-based approach to release planning works well
  - It provides enough planning to make progress on the project
  - It lacks enough detail to avoid giving a false sense of accuracy
    - People will be aware that there can be errors made in the estimates and can react once new information is available to make the errors clear
- In iteration planning, you need to engage in more detail to help create accurate work plans over the days allocated to an iteration
  - An iteration planning meeting occurs “between iterations”
    - If it occurs “during” an iteration, then you need to include the time spent on it in your other estimates (perhaps by adjusting your velocity down by a point or two to account for it)

Iteration Planning (II)

- All developers and the customer/user must be present for an iteration planning meeting
  - The developers are required to help identify tasks and make estimates
  - The customer/user is required to answer questions about the stories
- The process involves
  - For each story in the iteration
    - engage in Q&A with customer/user about the story
    - convert story into tasks that need to be completed to finish the story
    - assign each task to a single developer
  - Each developer then estimates each assigned task; performs sanity check
    - if a developer is overloaded, rebalancing or more planning is needed

Tasks

- Task identification takes a story that is written in a customer perspective and transforms it into a set of steps that are written from a developer's perspective (finally!)
- "A job seeker can search for jobs" might be transformed into
    - Code basic search interface
    - Write controller to handle submissions from search interface and perform the search
        - Ensure that controller can access the database correctly
    - Write a view that will display the results
- Working on this step will require "design thinking" either to come up with an initial design for a system or to integrate this feature into the existing design

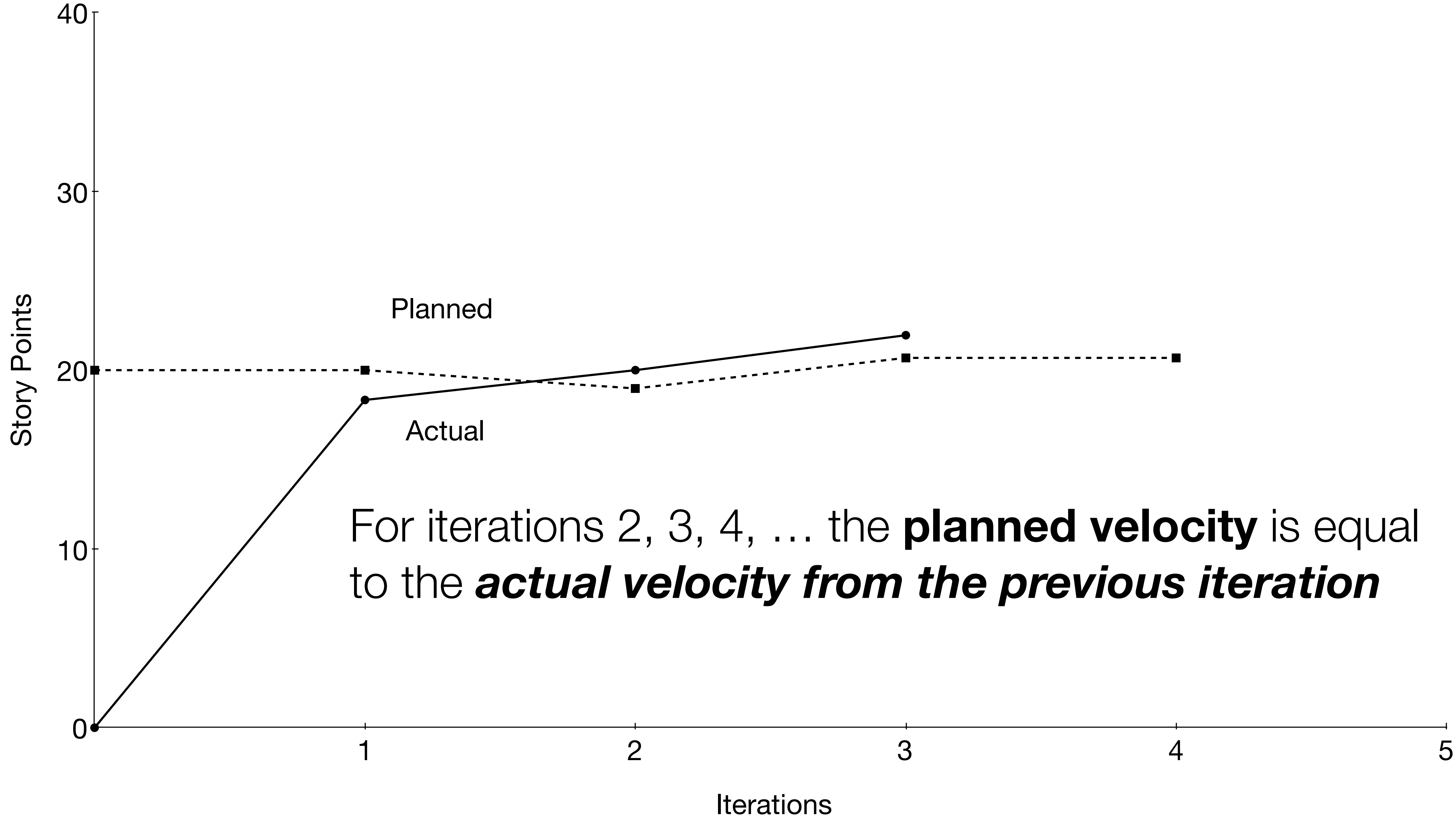
Task Estimation

- In release planning, we worked with “ideal days”
    - With task planning, we work with “ideal hours”
- Once a developer has their assigned tasks, they estimate the number of hours it will take to complete each one
    - They then add those hours up to perform a sanity check
    - They can also include a factor to transform ideal hours into actual hours
- Sanity Check
    - Compare number of hours with the length of the iteration
    - If the number of hours to complete the tasks is greater than the number of available hours, then rebalancing is needed
- A team perspective is needed to make this successful

Measuring and Monitoring Velocity

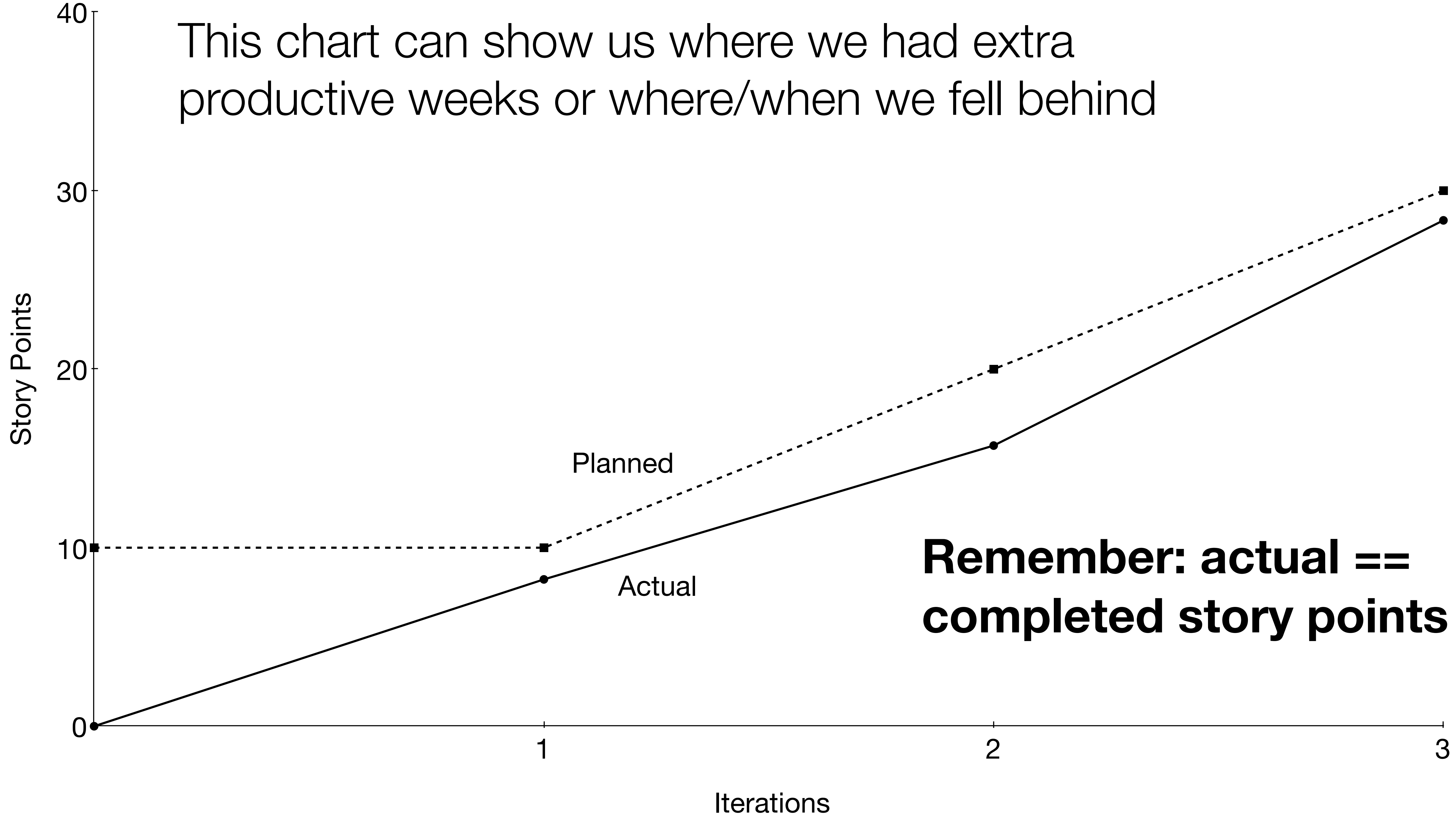
- Once points/priorities have been assigned and releases and iterations have been planned, the most important metric for an agile life cycle is velocity
    - velocity tracks how much work is completed in an iteration
        - before the iteration it is a "guess"
            - a guess that we have increased confidence in over time
        - after an iteration it is an actual metric that can be used in assessment
- How do we measure velocity?
    - The number of points associated with **completed** stories
        - Incomplete stories are not included (***velocity is an integer not a float***)
- With velocity measured, we can chart our progress in a variety of ways

Planned vs. Actual Velocity



For iterations 2, 3, 4, … the **planned velocity** is equal to the ***actual velocity from the previous iteration***

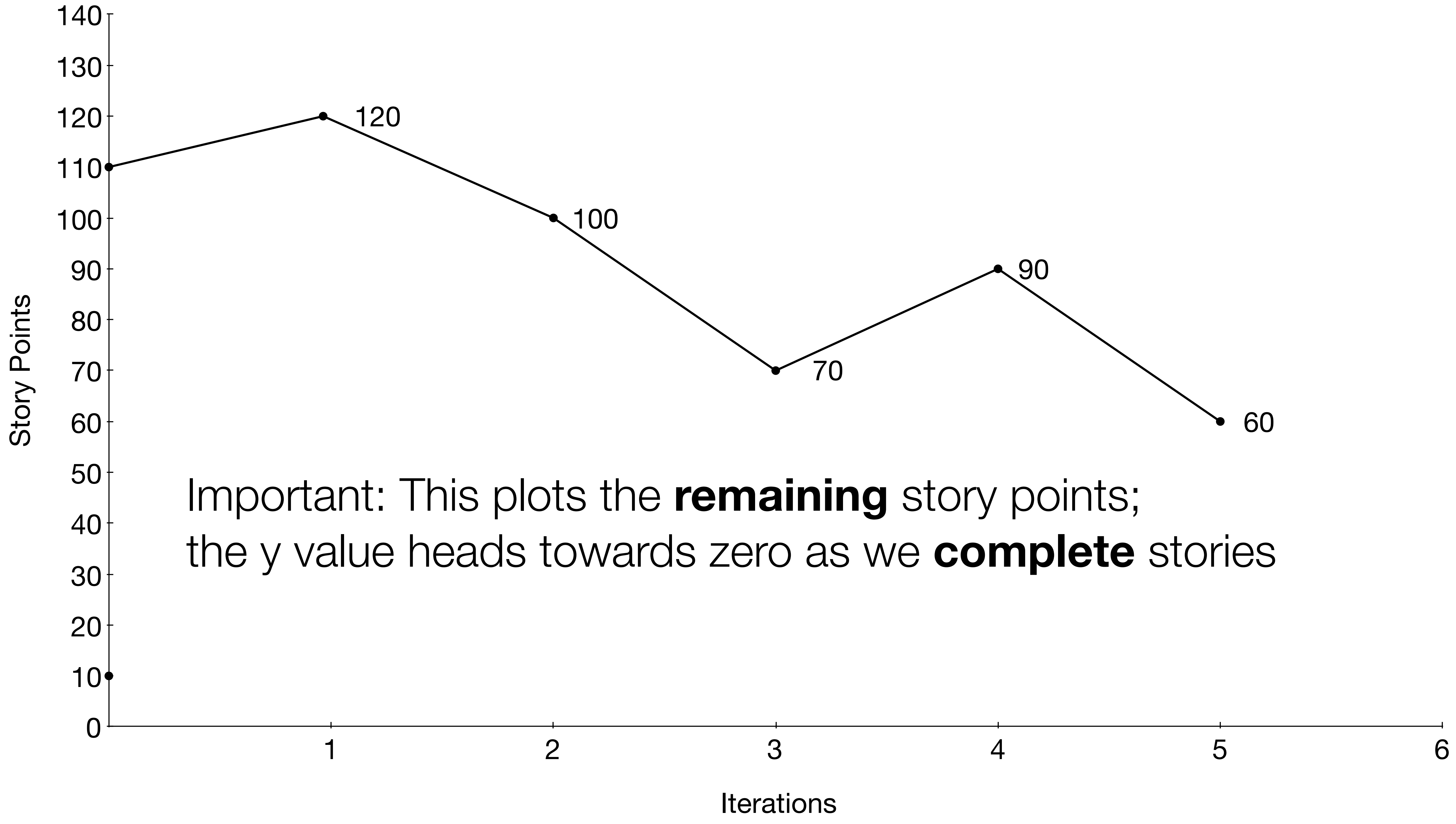
Planned vs. Actual Cumulative



This chart can show us where we had extra productive weeks or where/when we fell behind

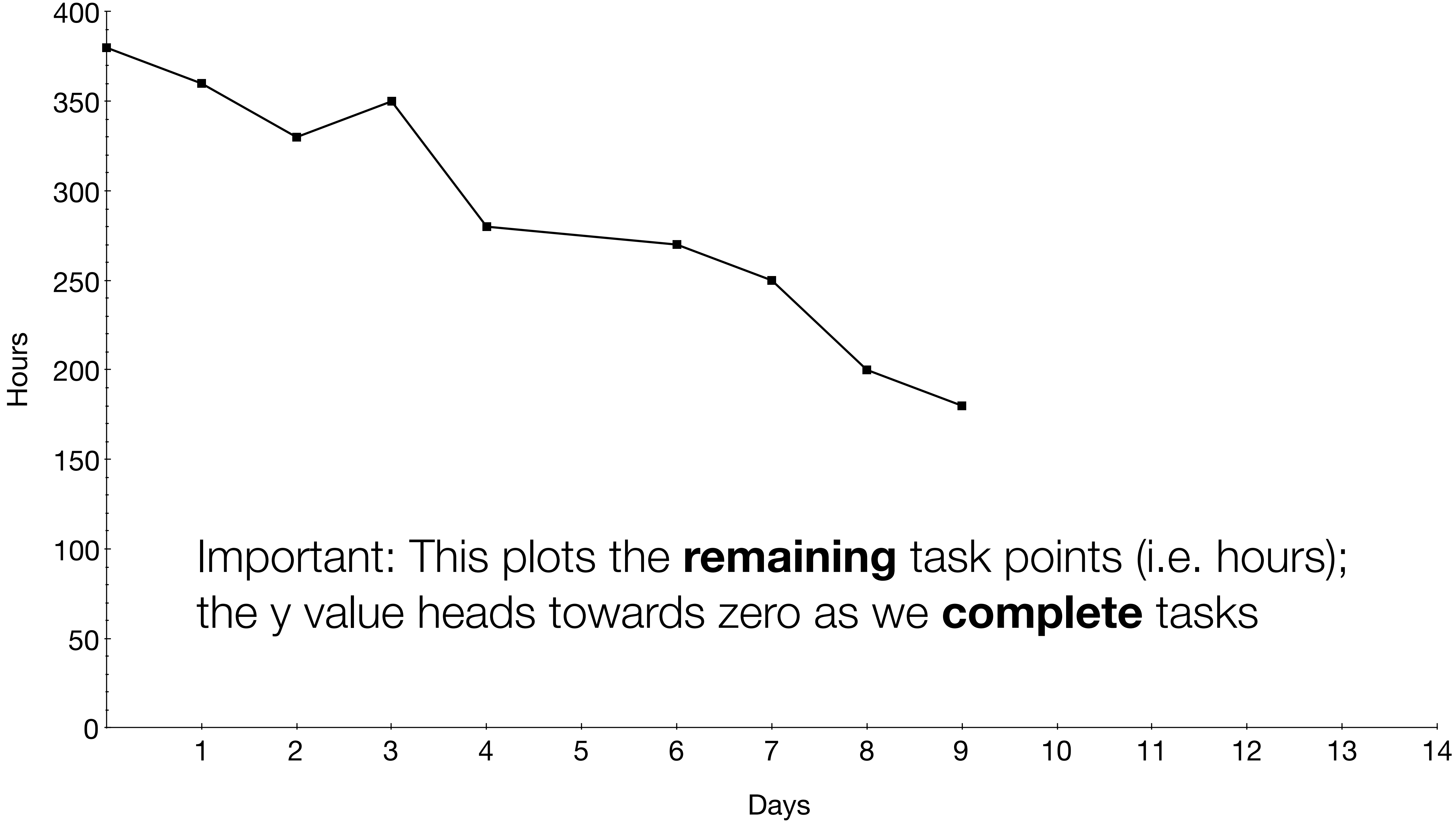
Remember: actual == completed story points

Iteration Burndown Charts



Important: This plots the **remaining** story points;
the y value heads towards zero as we **complete** stories

Daily Burndown Charts



Important: This plots the **remaining** task points (i.e. hours); the y value heads towards zero as we **complete** tasks

Summary

- In executing an agile life cycle, you must
    - estimate your stories
    - plan your releases
    - plan your iterations
    - measure your progress
- We have looked at various recommendations for performing these tasks
    - using “ideal days” (stories) and “idea hours” (tasks) for estimates and then using a conversion factor to get to “actual days” and “actual hours”
    - saw example charts to measure actual progress
        - Agile life cycles are brutal; if you fall behind, you’ll know it fast
            - the good news is that you’ll deal with schedule delays quickly and hopefully before they become a problem