CS 4530: Fundamentals of Software Engineering
Module 6.2: Agile Planning and Estimation

Adeel Bhutta, Jan Vitek and Mitch Wand

Khoury College of Computer Sciences

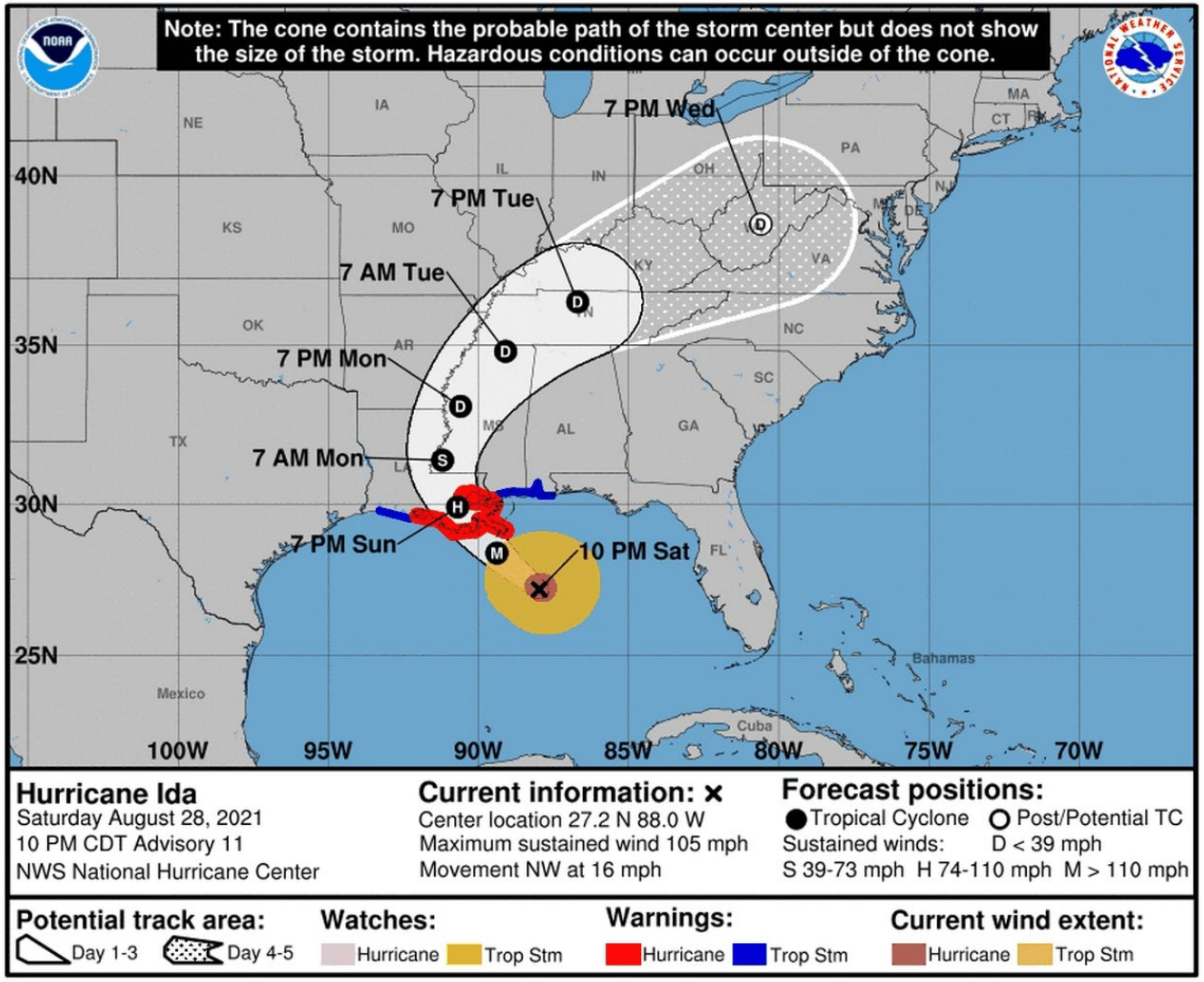
Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Describe how agile planning manages uncertainty by creating detailed plans only for the most immediate tasks
  - Explain how agile planning decomposes large projects into individual tasks that can be estimated
  - Understand the key artifacts and process steps in Scrum

Requirements: Which to pick?

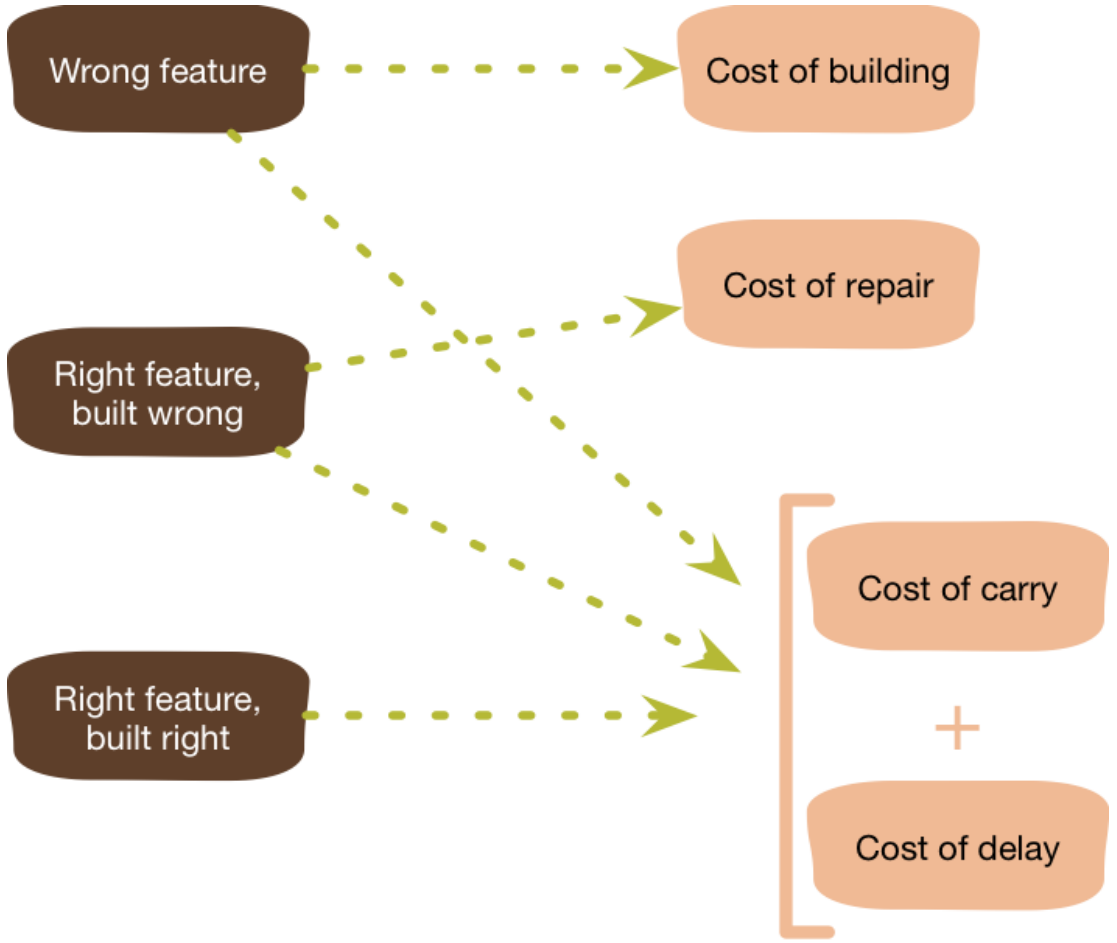
- There are four knobs you can adjust when negotiating requirements:
  - Project scope
  - Project duration
  - Project quality
  - Project cost
- Usually cost is most constrained: you have a budget to spend, and you have a headcount of developers to pay
- Determining feasible scope, timeline and maximizing quality is the subject of much software engineering research

Lesson from Meteorology: Uncertainty in Estimation



Agile Principles for Effective Planning: YAGNI

- YAGNI – "You Aren't Going To Need It"
- Do not pre-maturely plan or implement features
- Why? Uncertainty *in what we actually need*
- Focus on *prioritization*, independent of estimation



Graphic: Martin Fowler

Tracking and Prioritizing Tasks: Product Backlog

- List of user stories for the product
- All entries should add value
- No low level tasks

PlowTracker Product Backlog

Item	Priority	Value
The driver's interface should display unplowed streets	High	Required for MVP – drivers must know where to go
The driver's interface should track which streets have been plowed	High	Required for MVP – informs rest of system what has been plowed
The city official's internal interface should show estimated arrival times for plows	Medium	City officials field thousands of complaint calls, expected to increase citizen satisfaction
The driver's interface should show an optimized plowing route	Low	Plowing will become more efficient, cuts fuel and labor costs
Members of the public should be able to see real-time plow status	Low	Government transparency groups want this; it might reduce phone complaints

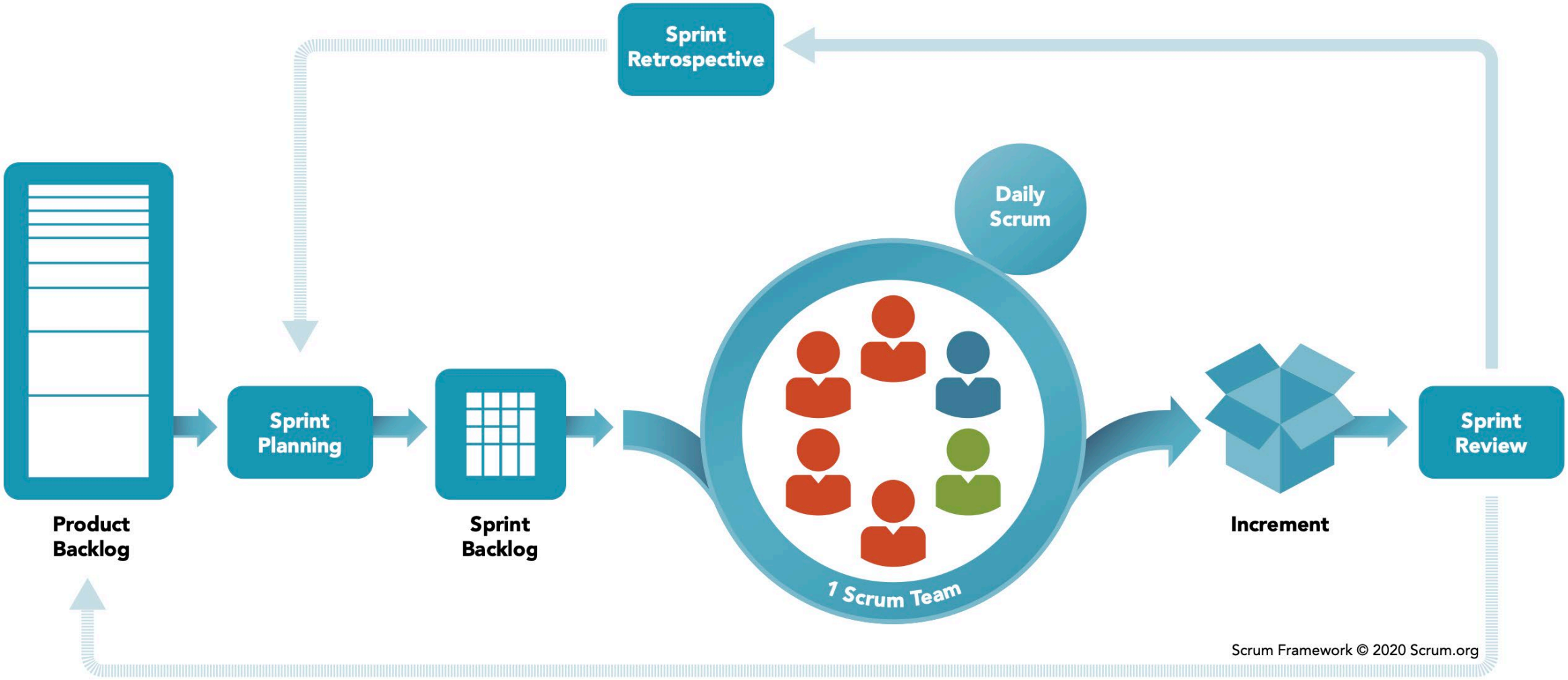
Tracking and Prioritizing Tasks: Product Backlog

- List of user stories for the product
- All entries should add value
- No low level tasks
- Items are prioritized
- A living document

PlowTracker Product Backlog

Item	Priority	Value
The driver's interface should display unplowed streets	High	Required for MVP – drivers must know where to go
The driver's interface should track which streets have been plowed	High	Required for MVP – informs rest of system what has been plowed
The city official's internal interface should show estimated arrival times for plows	Low	City officials field thousands of complaint calls, expected to increase citizen satisfaction
The driver's interface should show an optimized plowing route	Low	Plowing will become more efficient, cuts fuel and labor costs
Members of the public should be able to see real-time plow status	High	Government transparency groups want this; it might reduce phone complaints

Scrum: Daily progress towards product goals



Planning a Sprint

- Select user stories for the sprint **based on** priority and value
- Decompose stories into detailed tasks
- Estimate duration of each task (max 1 day each)
- Time-boxed meeting – don’t make every decision here
- Include non-story tasks as needed (e.g. quality improvements, knowledge acquisition)

Planning a Sprint Backlog

- Sprint Focus:
  - “The driver’s interface should display unplowed streets”
  - “The driver’s interface should track which streets have been plowed”
- Sprint tasks:
  - Tasks for API design
    - Work out the interface for CRUD on plowed streets
  - Tasks for app development
    - Design the interface for viewing unplowed streets
    - Create the map interface that shows streets in the city
    - Fetch unplowed streets from API and update the map
    - Update the API with current location while plowing in progress
  - Tasks for backend development
    - Determine how to model and store plowed street data
    - Implement tests for expected API behavior
    - Implement API to mark street as plowed
    - Implement API to fetch unplowed streets

Estimating Task Time

- Collaborative team process:
  - Estimate how long each task will take (hard)
  - Estimate relative size of each task (easier)

Estimating with T-Shirt Sizes

Example: Estimating with T-Shirt

Discussion: Would you accept these estimates? Why? What factors should the team consider? Are tasks missing?

Task	Size	Rationale
Work out the interface for CRUD on plowed streets	Medium	We've designed similar APIs before, there will likely be some new aspects, but it will not be too hard. It will require coordination.
Design the interface for viewing unplowed streets	Small	The client has already shared mockups of what they want
Create the map interface that shows streets in the city	Large	We haven't worked with mapping APIs before; maybe we should decompose this into smaller tasks, there is risk here.
Fetch unplowed streets from API and update the map	Medium	We think that this should be easy, it's just patching a few components together, but don't yet know enough about how the map will be implemented to know for sure how hard this is.
Update the API with current location while plowing in progress	Small	We know exactly what API call to make here, and how to fetch location, it's easy

Planning helps us find what we don’t know

Task	Size	Rationale
Research OpenStreetMap API and examples of its usage	Small	We’ve learned how to use other APIs before. This one looks well documented, and spending a few hours looking at examples will probably go a long way.
Create OpenStreetMap prototype, showing a static map with streets	Small	A quick internet search shows plenty of examples, this should be easy to adapt
Create the map interface that shows streets in the city	Medium	Once we’ve built a throwaway prototype that shows the city map, we can leverage that knowledge to build the map in our app

”Sprint 0” Tasks to Help Estimate Stories

- Find resources to gain more experience about a technology or about a problem domain
- Create prototypes that you can throw away
- Consider having multiple developers implement different approaches
- Create load tests/simulations to identify the performance limits of technology or architecture
- Learn just enough to make a *responsible* estimate

Daily Scrum

- 15 minutes maximum “stand up” meeting
  - What have I done?
  - What am I working on?
  - What am I stuck on/need help on?
- Conversation focuses on goals:
  - Transparency between team members
  - Encourage adaptation

Sprint Review and Retrospective

- Sprint Review:
  - Provide a working demo
  - What did we get done?
  - What value did we deliver?
- Sprint Retrospective
  - What went well?
  - What could we have done better?
  - If incidents occurred: conduct a blameless postmortem
- Provides an opportunity to reflect on overall project velocity

Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Describe how agile planning manages uncertainty by creating detailed plans only for the most immediate tasks
  - Explain how agile planning decomposes large projects into individual tasks that can be estimated
  - Understand the key artifacts and process steps in Scrum