Agile Software Development for Developers

Session 7: Sprint Execution

Yousef Mehrdad Bibalan
www.bibalan.com

Review

- Session 1
  - Paradigm and paradigm shift
  - Agility: An elephant in the dark
  - Agility: A definition
  - Agile values, principles, and practices
  - The Cynefin: clear, complicated, complex, chaotic, disorder
- Session 2
  - Product backlog items: feature (user story), defects, technical work, and knowledge acquisition (spike)
  - User story: title, description, acceptance criteria
  - Questions words: who, what, why
  - User story is something like order
- Session 3
  - Estimation: what and when
  - Estimation: Basic concepts
  - Estimation: Product backlog estimation concepts

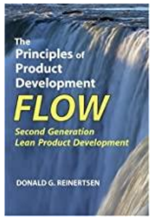
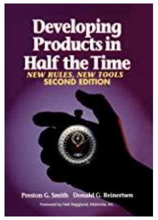
Review

- Session 4
  - PBI Estimation Units
  - Estimation Scale
  - Planning Poker
- Session 5
  - Velocity in Physics
  - Velocity for An Agile Team
  - Velocity like Stock Price
  - Yesterday’s Weather
  - Release Planning (Fixed Scope)
  - Exercise: Story points are influenced by
    - The Amount of Work
    - Uncertainty and Risk
    - Complexity
    - Definition of Done

Review

- Session 6
  - Velocity-Driven vs. Capacity-Driven Sprint Planning
  - Velocity-Driven Planning
  - Capacity-Driven Sprint Planning
  - Capacity-Driven Sprint Planning: The Approaches
    - Two-Part Sprint Planning
    - One-Part Sprint Planning
    - Determine Team Capacity
      - Time Needed for Sprint Activities
      - Tools Support
  - Sprint Planning Tips
    - Do We Assign Team Members to Tasks?
    - Size of Tasks
    - Sprint Planning: Estimation, Commitment or Guarantee

Many developers incorrectly assume that the sooner they start work, the sooner they will finish it.
— Don Reinertsen


The Principles of Product Development
FLOW
Second Generation Lean Product Development
DONALD G. REINERTSEN

Developing Products in Half the Time
NEW RULES, NEW TOOLS
SECOND EDITION
Preston G. Smith Donald G. Reinertsen

It's not "the more you start, the more you finish," it's "the more you finish, the more you finish."
—David P. Joyce

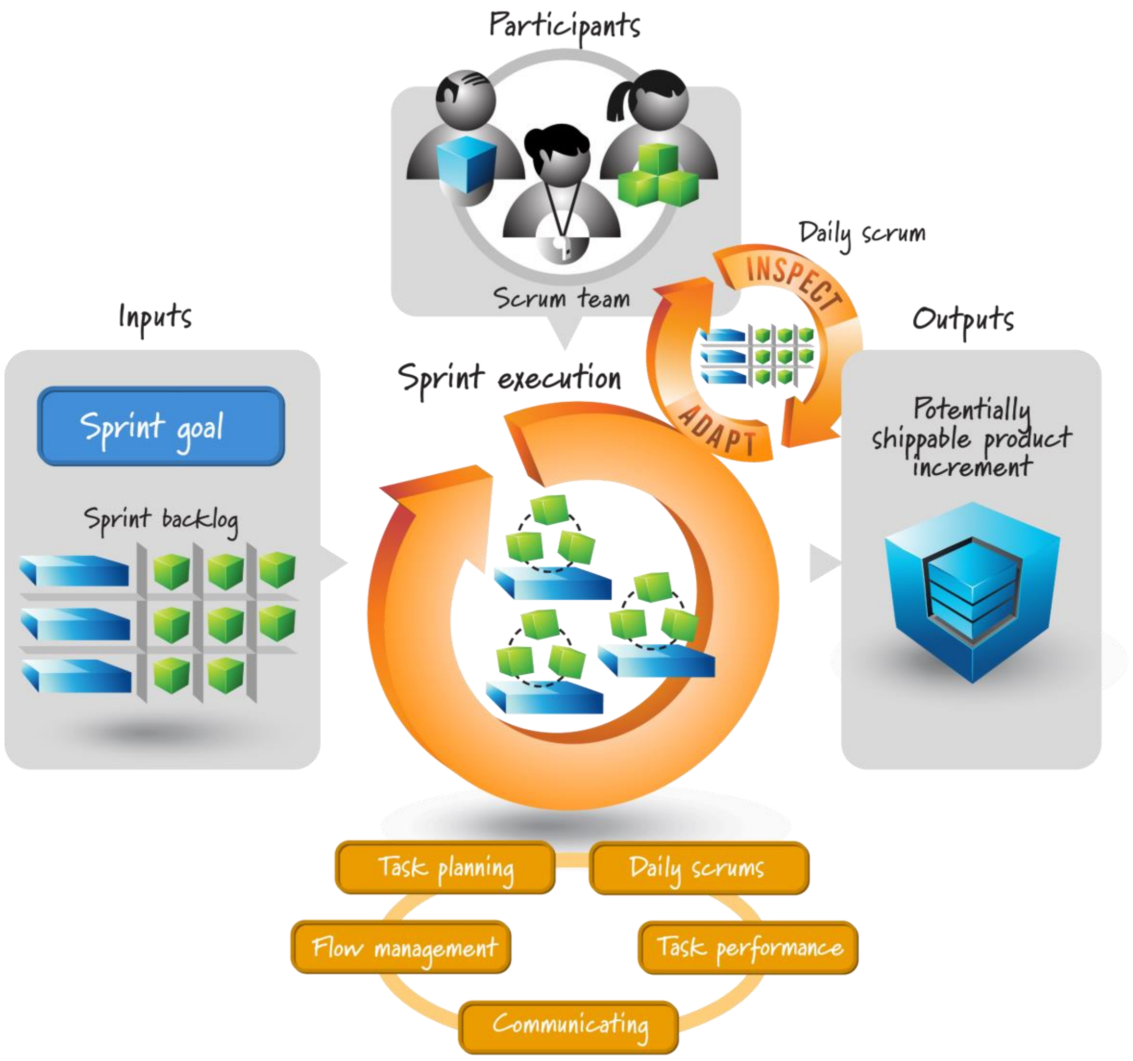
Sprint Execution

Context and Approaches

Sprint Execution: Team!

- The Product Owner
  - The Product Owner must be available during sprint execution to:
  - Answer clarifying questions
  - Review intermediate work and provide feedback to the team
  - Discuss adjustments to the sprint goal if conditions warrant
  - Verify that the acceptance criteria of product backlog items have been met.
- The Scrum Master
- The Development Team
  - The development team members self-organize and determine the best way to meet the goal established during sprint planning.

Sprint Execution



Sprint Execution

Task Planning

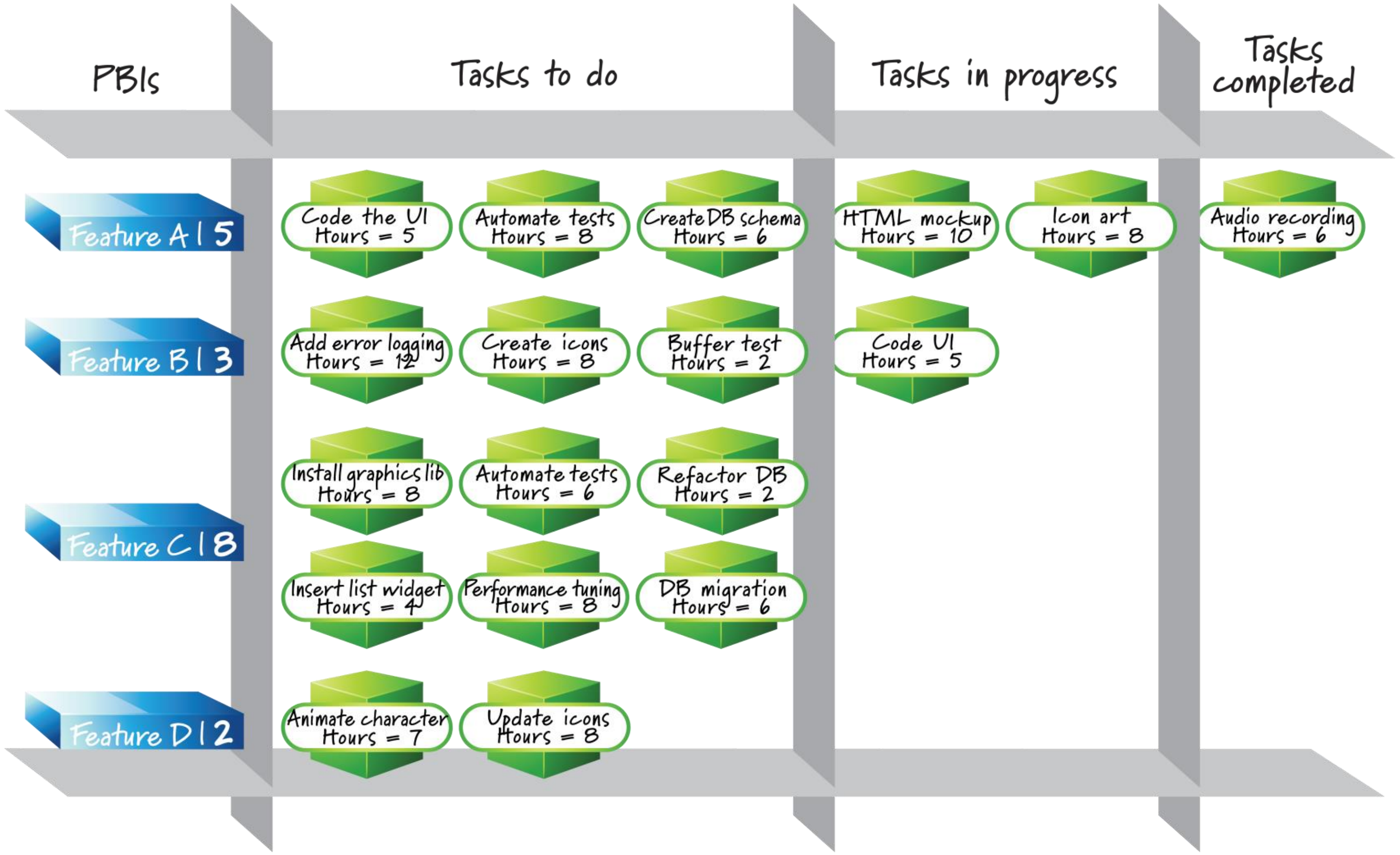
Task Planning

- Who assigns the tasks?
  - The ScrumMaster doesn’t assign work to the team or tell the team how to do the work.
  - A self-organizing team figures these things out for itself.
- We don’t need Gantt chart or etc.
  - The size of team is small (about 7± 2)
  - The duration is almost two weeks
- Task planning
  - Some up-front planning is helpful for exposing important task-level dependencies in Sprint Planning
  - Allow task planning to occur continuously during sprint execution

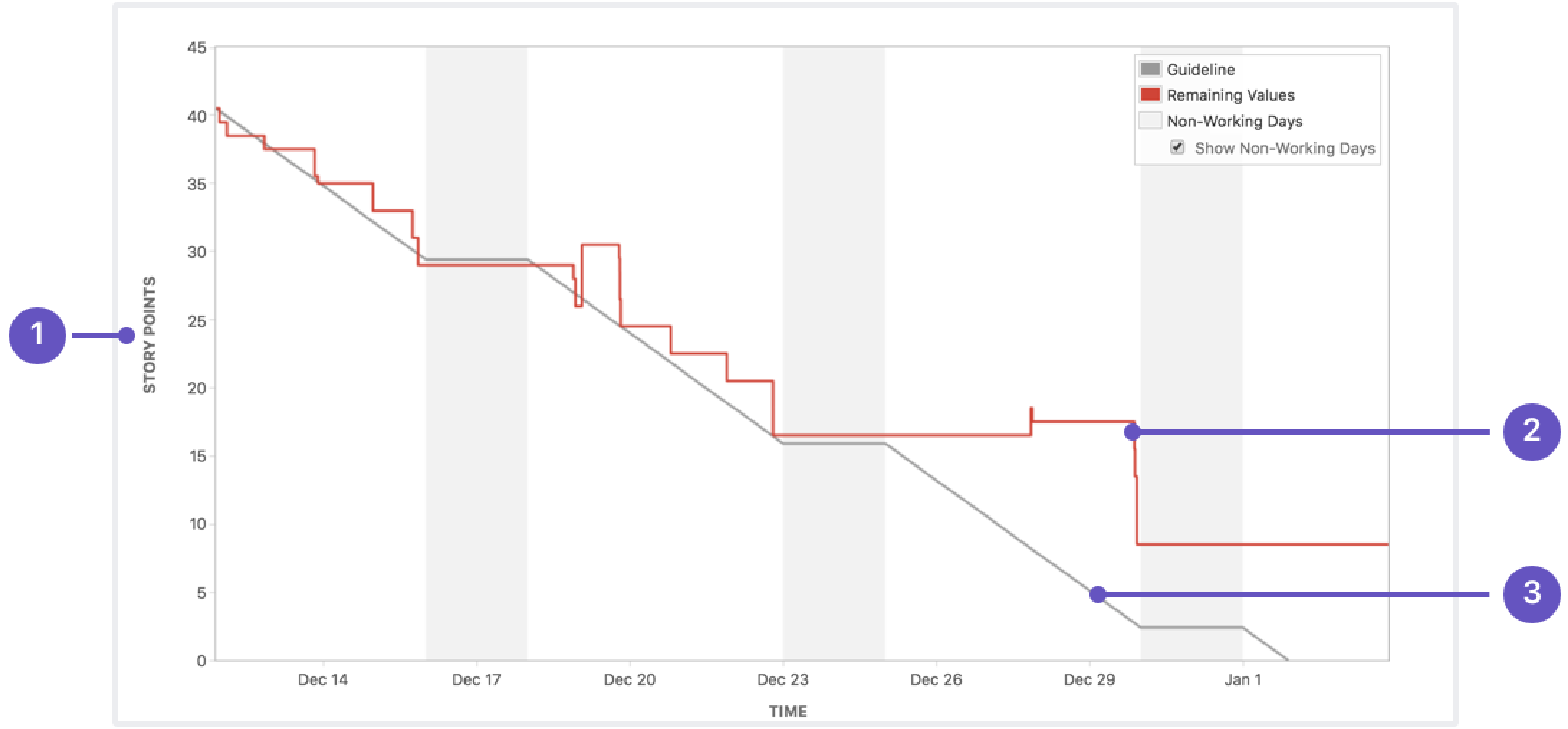
Sprint Execution

Communicating

Task Board (Sprint Backlog)



Sprint Burndown Chart

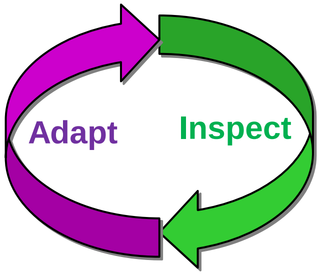


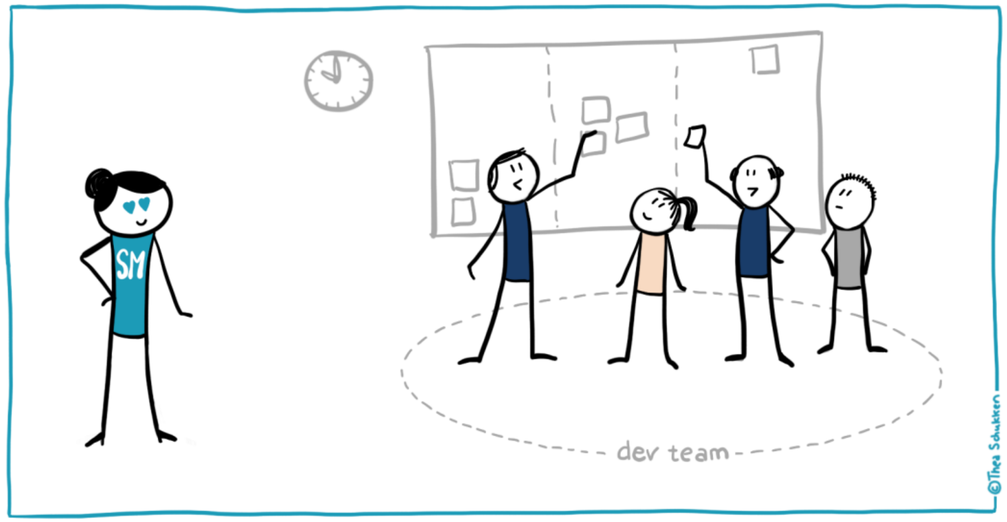
Jira Sprint Burndown Chart

Sprint Execution

Daily Scrum

Daily Scrum





- What did I accomplish since the last daily scrum?
- What do I plan to work on by the next daily scrum?
- What are the obstacles or impediments that are preventing me from making progress?

Daily Scrum: Why?

How does a project get to be a year late? … One day at a time.

The Mythical Man-Month,1995, Page 153

Frederick Phillips Brooks, Jr. is a **computer architect, software engineer**, and **computer scientist**, most famous for managing the development of IBM's System/360 Computer family hardware and then OS/360

Brooks received a **Turing Award** in 1999 and many other awards.

Sprint Execution

Flow Management

Flow Management

Parallel Work

Parallel Work

- The team must determine how many product backlog items to work on in parallel (at the same time)
- Working on too many items at once slows the team down.
- Working on too few items at once is equally wasteful.

- WiP Limits: The Magic Sword
  - Work in process is the number of work items you have going at the same time.
  - The number of works that team has started but not finished yet.
  - The number of task items that a team is currently working on.

The Importance of WiP Limit: Little's Law and WiP

- Little's Law

$$\overline{\text{Lead Time}} = \overline{WiP} \;/\; \overline{\text{Delivery Rate}}$$

$$\overline{\text{Lead Time}}\downarrow \quad \Rightarrow \quad \overline{WiP}\downarrow \qquad or \qquad \overline{\text{Delivery Rate}}\uparrow$$

In order to decrease the Lead Time for work items, we must decrease (limit) the Work in Progress.

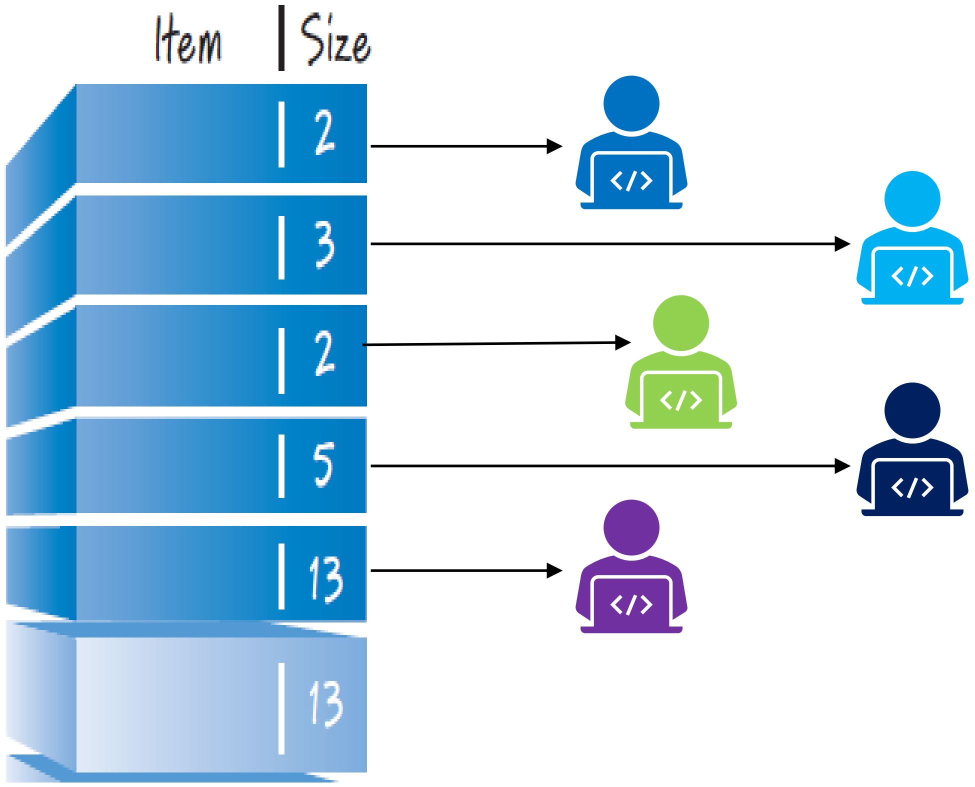
Flow Management

Swarming

Swarming

- A behavior whereby team members with available capacity and appropriate skills collectively work (**swarm**) on an **item** to finish what has already been started before moving ahead to begin work on new items.

Swarming



≠

Magic of Managers by Assigning Tasks:

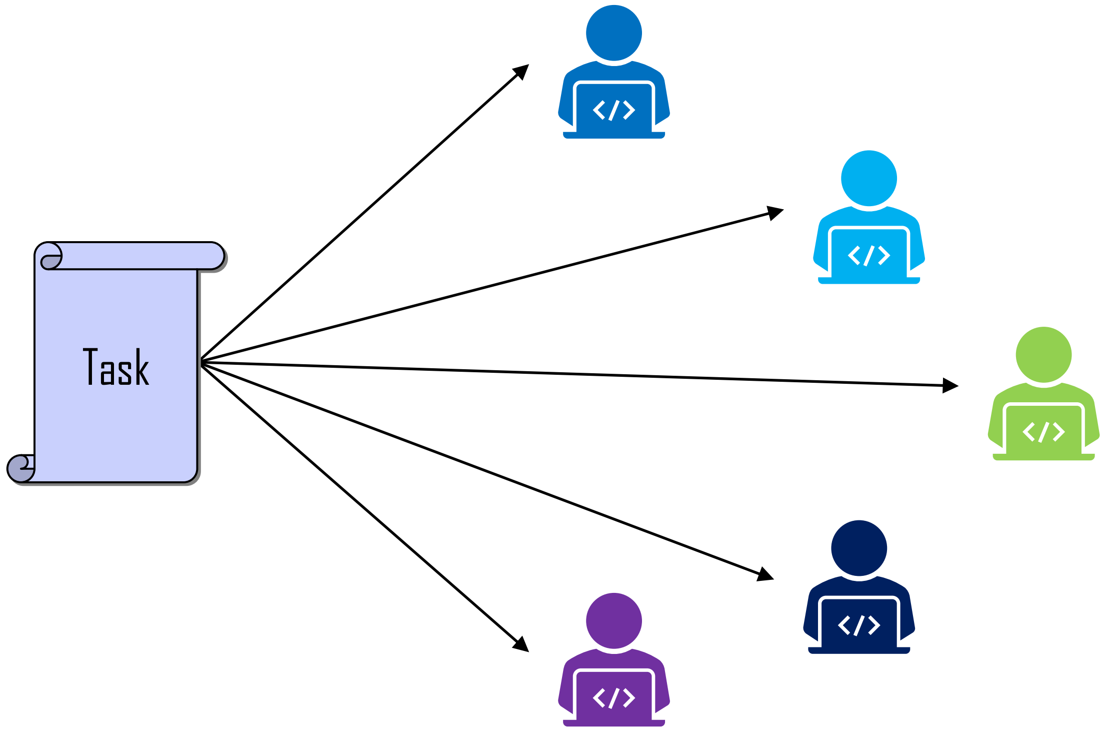
1 Team of 5 ==> 5 Teams of 1

Credit: Essential Scrum

Flow Management

More …

Who Does the Work?

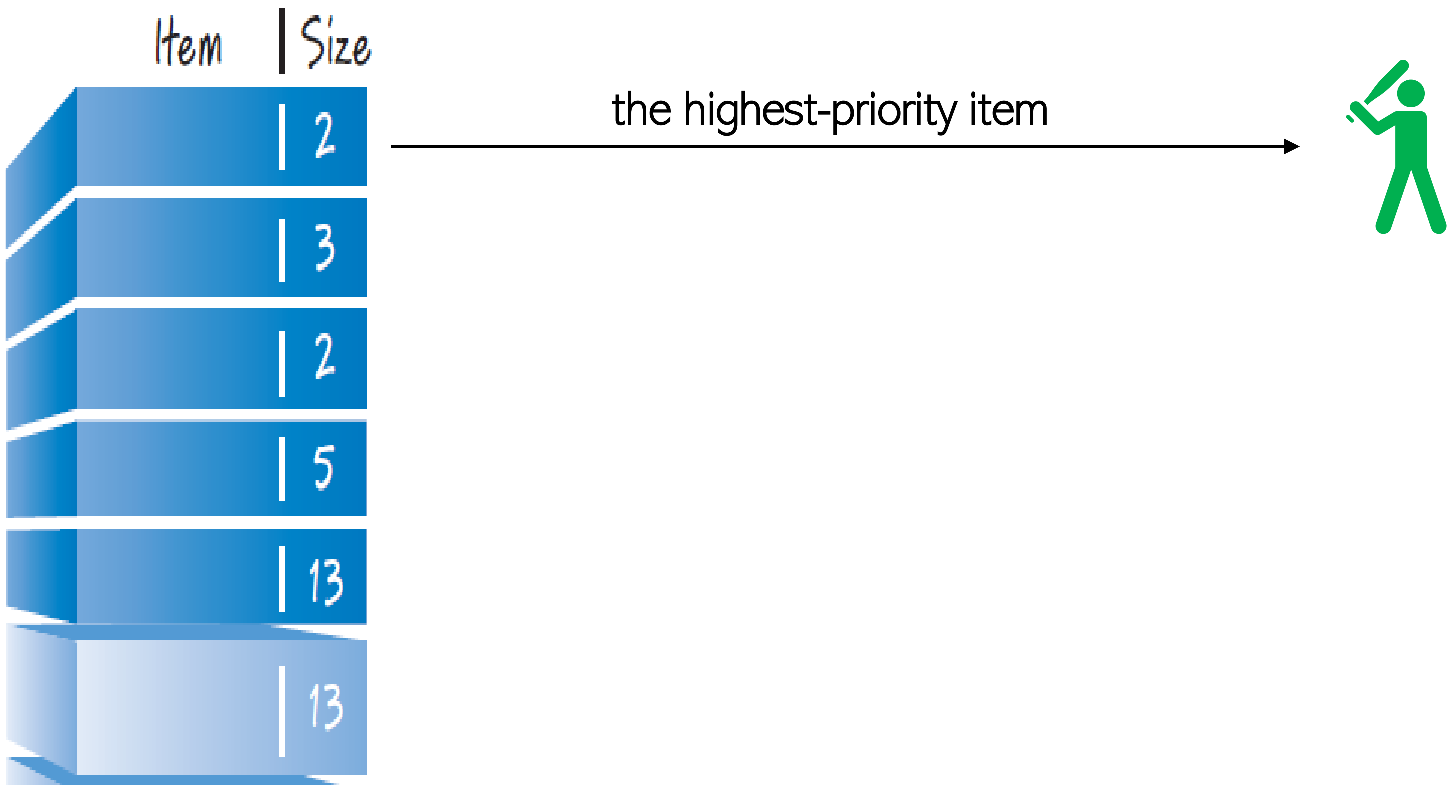


Answer: the person best able to quickly and correctly get it done.

And if that person is unavailable?
Answer: The team should decide on the next best person.

Credit: Essential Scrum

Which Items to Start



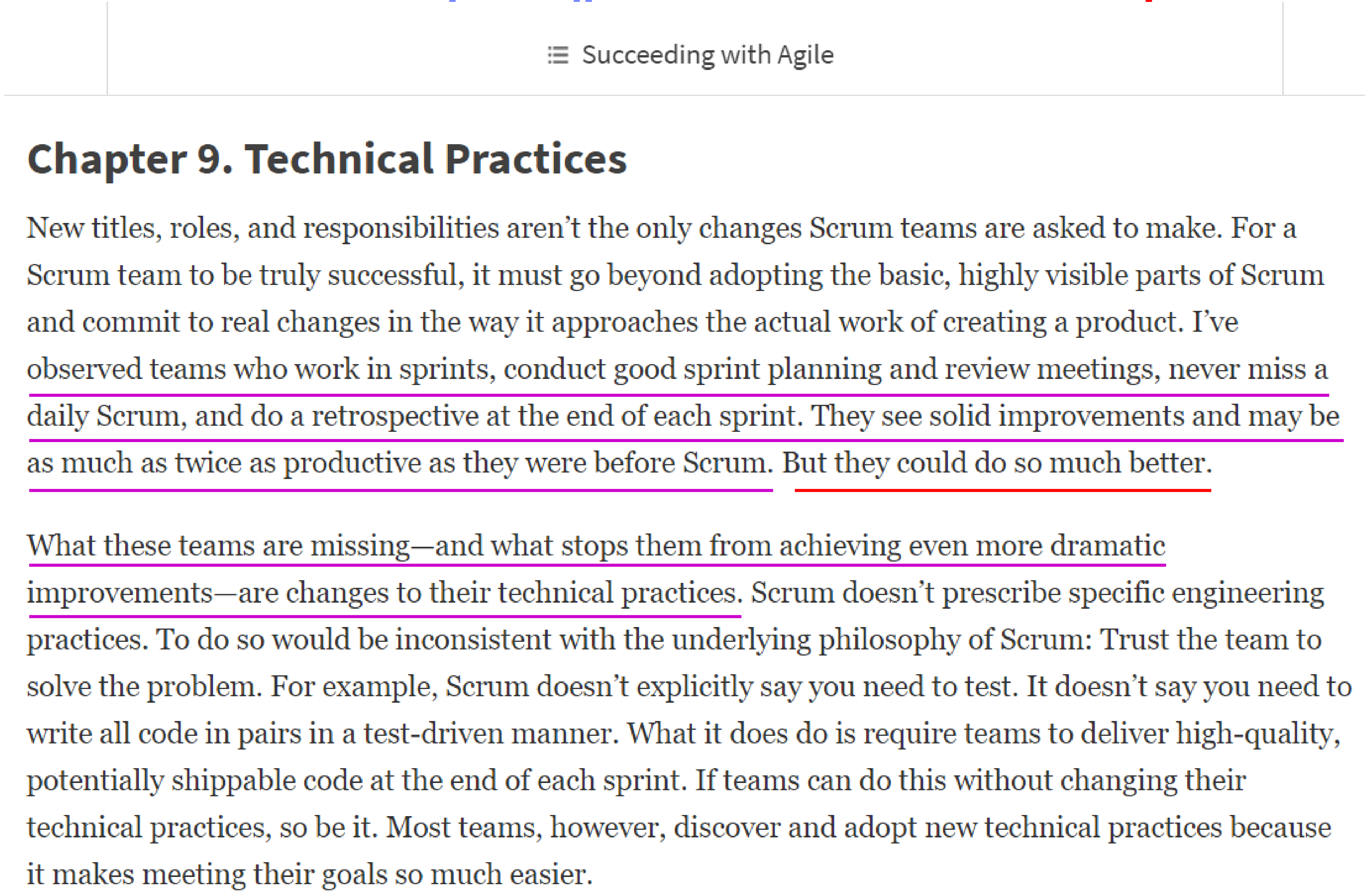
Dependencies or skills capacity constraints might dictate a different order.

Credit: Essential Scrum

Sprint Execution

Task Performance

Technical Practices: Improving Technical Practices is not Optional



Succeeding with Agile

Chapter 9. Technical Practices

New titles, roles, and responsibilities aren't the only changes Scrum teams are asked to make. For a Scrum team to be truly successful, it must go beyond adopting the basic, highly visible parts of Scrum and commit to real changes in the way it approaches the actual work of creating a product. I've observed teams who work in sprints, conduct good sprint planning and review meetings, never miss a daily Scrum, and do a retrospective at the end of each sprint. They see solid improvements and may be as much as twice as productive as they were before Scrum. But they could do so much better.

What these teams are missing—and what stops them from achieving even more dramatic improvements—are changes to their technical practices. Scrum doesn't prescribe specific engineering practices. To do so would be inconsistent with the underlying philosophy of Scrum: Trust the team to solve the problem. For example, Scrum doesn't explicitly say you need to test. It doesn't say you need to write all code in pairs in a test-driven manner. What it does do is require teams to deliver high-quality, potentially shippable code at the end of each sprint. If teams can do this without changing their technical practices, so be it. Most teams, however, discover and adopt new technical practices because it makes meeting their goals so much easier.

Five Common Practices (1)

- Test-Driven Development
  - (1994) Extreme Programming: “we usually write the test first”
  - (1998 to 2002): “Test First” is elaborated into “Test Driven”
  - (2006 -..): further innovations derived from TDD such as ATDD or BDD

- Refactoring
  - Refactoring consists of improving the internal structure of an existing program’s source code, while preserving its external behavior.
  - Refactoring does “not” mean:
    - rewriting code
    - fixing bugs
    - improve observable aspects of software such as its interface

- Pair Programming
  - Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair).
  - The programmer at the keyboard is usually called the “driver”, the other, also actively involved in the programming task but focusing more on overall direction is the “navigator”; it is expected that the programmers swap roles every few minutes or so.

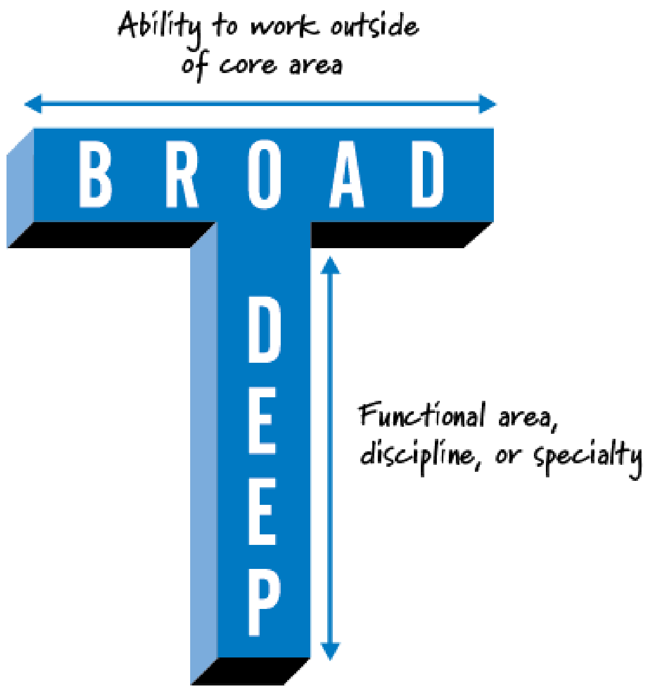
Five Common Practices (2)

- Collective Ownership
  - Collective ownership refers to all developers feeling ownership over all artifacts of the development process, but especially of the code and automated tests.
  - Collective ownership is not intended to cause a free-for-all in the coding. Programmers will still tend to have certain areas they specialize in and prefer to work in.
  - everyone on the team shares the following responsibilities:
    - Ensure that no developer becomes so specialized he can contribute only in one area.
    - Make certain that no area becomes so intricate that it is understood and worked upon by only one developer.

- Continuous Integration
  - A technical practice where members of a single team or multiple teams integrate their work as frequently as is practical.
  - Two objectives:
    - minimize the duration and effort required by each integration episode
    - be able to deliver a product version suitable for release at any moment

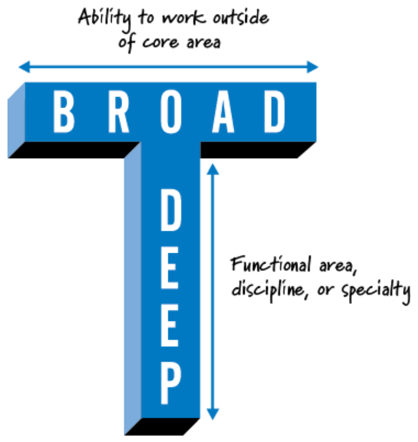
T-Shaped Skills

- A metaphor used to describe a person with deep vertical skills in a specialized area (such as UX design) as well as broad but not necessarily very deep skills in other relevant areas (such as testing and documentation). Team members with T-shaped skills better enable swarming behavior. See also *swarming*.


Ability to work outside of core area
BROAD
DEEP
Functional area, discipline, or specialty

T-Shaped Skills vs Cross-Functional Team

- cross-functional team
  - A team composed of members with all the functional skills (such as UI designers, developers, testers) and specialties necessary to complete work that requires more than a single discipline.


Ability to work outside of core area
BROAD
DEEP
Functional area, discipline, or specialty

$\neq$

Cross-Functional Team

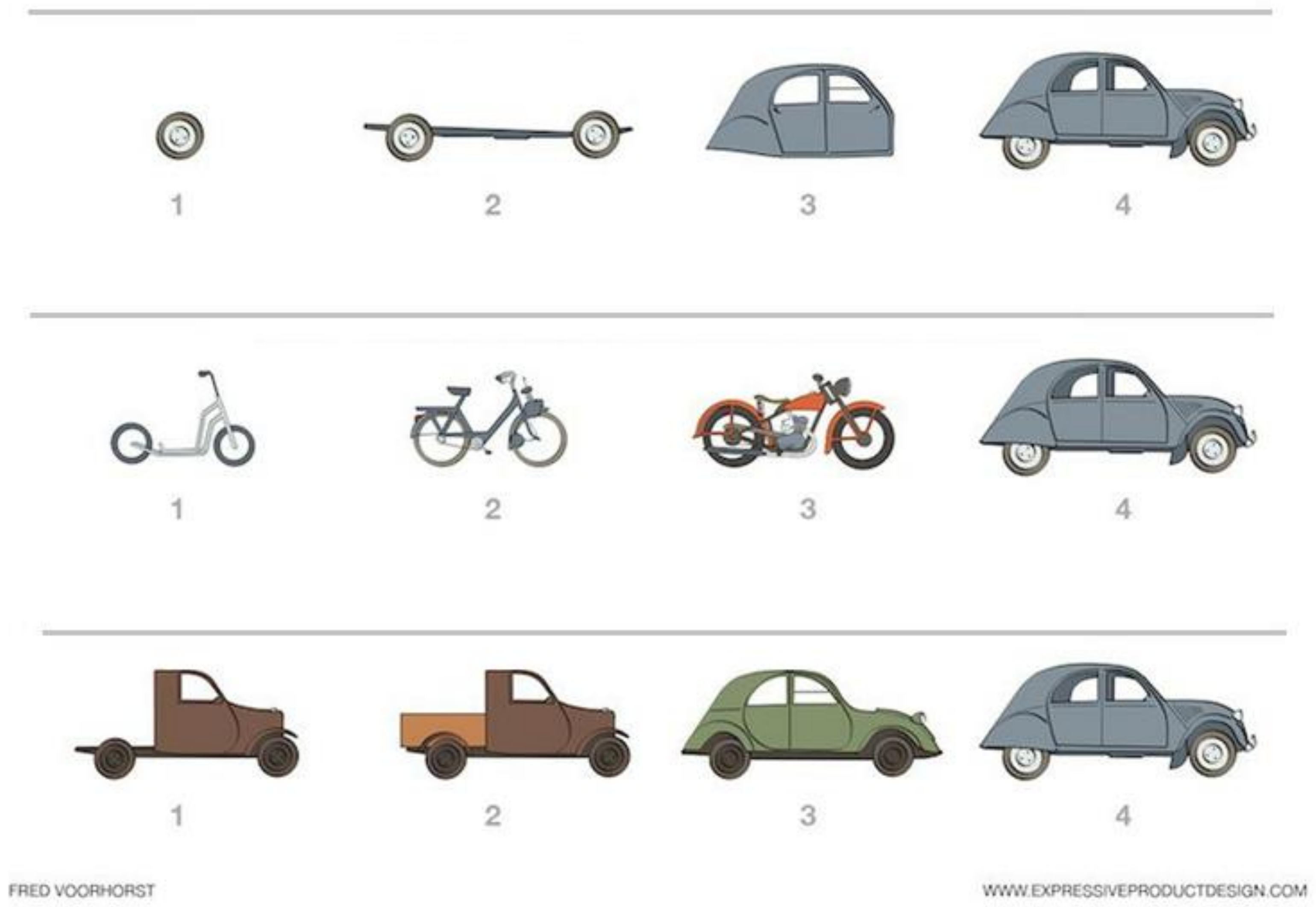
Wrap-Up

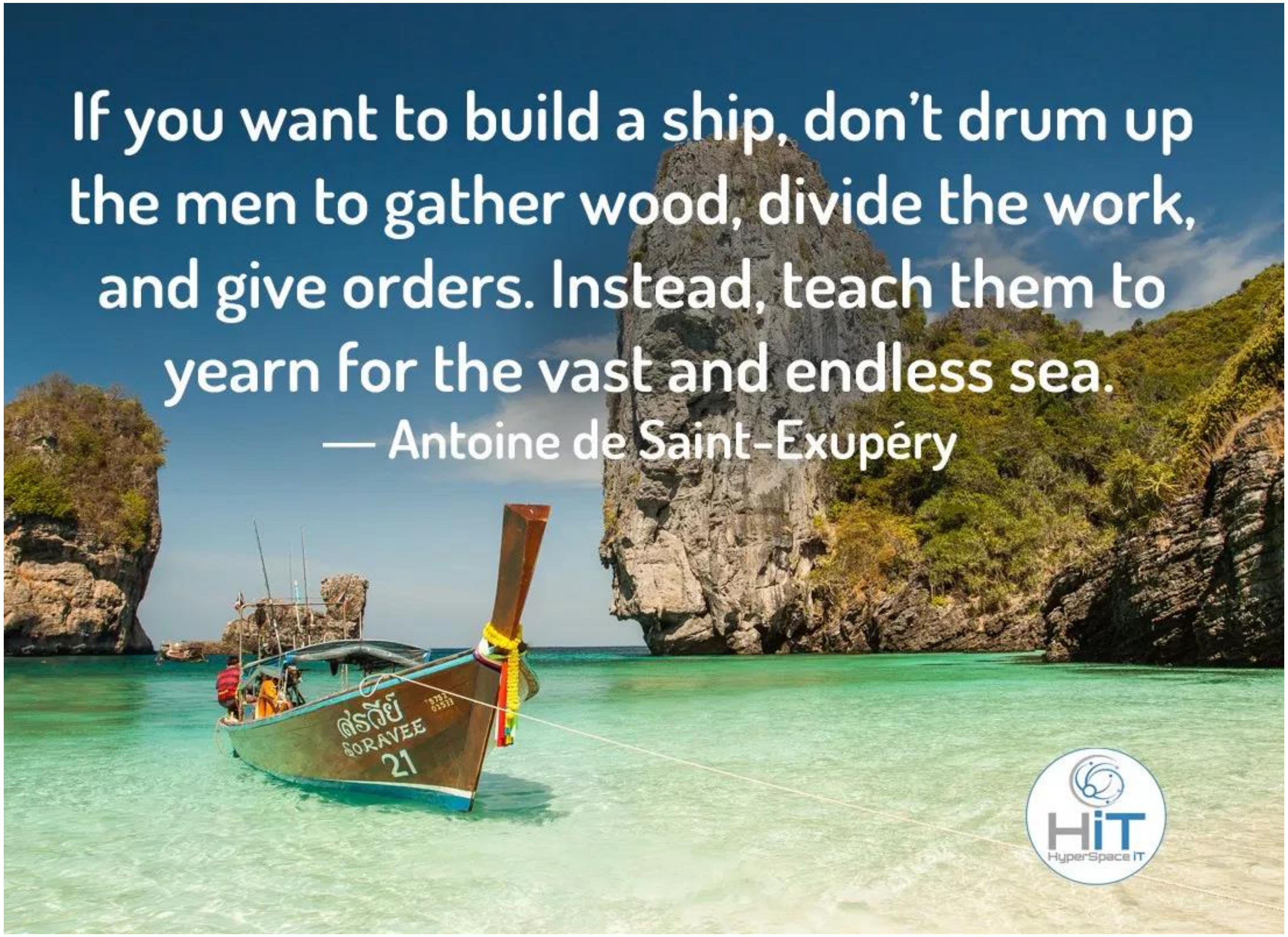
Which one is an agile way?

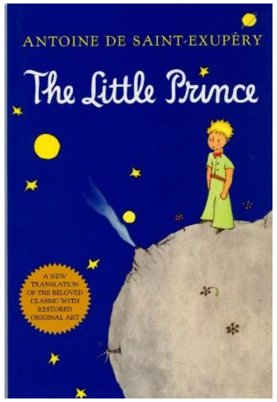
1

2

3




If you want to build a ship, don't drum up the men to gather wood, divide the work, and give orders. Instead, teach them to yearn for the vast and endless sea.
— Antoine de Saint-Exupéry
HiT
HyperSpace IT


ANTOINE DE SAINT-EXUPÉRY
The Little Prince

Thanks

- Faezeh Eshragh
- Reza Moghaddas Jafari
- Hossein Nassiri
- Reza Rahmati
- Mohammad Nadi