Process and estimation

Software engineering process:

Through what process do you make the software?

What’s the goal of the software engineering process for startups?

What’s the goal of the software engineering process for startups?

- Don’t spend money recklessly
- Make money by building something people want to buy

Make sure the company survives

How is the software engineering process different at e.g., Google or Microsoft?

“Agile” software development Prior slides say:

- A set of software engineering principles and best practices (allowing for rapid delivery of high-quality software)
- A business approach (aligning development with customer needs and goals)

“Agile” software development
Prior slides say:

- A set of software engineering principles and best practices (allowing for rapid delivery of high-quality software)
- A business approach (aligning development with customer needs and goals)

It’s a framework that historically works ok for startups
Having a framework is remarkably helpful

Let’s think through agile software development principles

Agile principles

- What is our highest priority?

Agile principles

- What is our highest priority?
- What if the customer changes their mind? What if we do more discovery and the requirements change?

Agile principles

- What is our highest priority?
- What if the customer changes their mind? What if we do more discovery and the requirements change?
- How often should we release new software?

Delivering software *for the customer*

Agile principles

- How should the “business side” and the “engineering side” interact?

Agile principles

- How should the “business side” and the “engineering side” interact?
- How should we measure progress?

Agile principles

- How should the “business side” and the “engineering side” interact?
- How should we measure progress?

Delivering software *for the customer*

Agile principles

- How should the “business side” and the “engineering side” interact?
- How should we measure progress?

- “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely”

Ok, team:
Do we care about writing good code?

Ok, team:
Do we care about writing good code?

What counts as good code?
Who owns the code?

Agile practice

Backlogs (Product and Sprint)	Behavior-driven development (BDD)	Cross-functional team	Continuous integration (CI)	Domain-driven design (DDD)	Information radiators (Kanban board, Task board, Burndown chart)	Acceptance test-driven development (ATDD)
Iterative and incremental development (IID)	Pair programming	Planning poker	Refactoring	Scrum meetings (Sprint planning, Daily scrum, Sprint review and retrospective)	Small releases	Simple design
Test-driven development (TDD)	Agile testing	Timeboxing	Use case	User story	Story-driven modeling	Retrospective
On-site customer	Agile Modeling	40-hour weeks	Short development cycles	Collective ownership	Open workspace	Velocity tracking

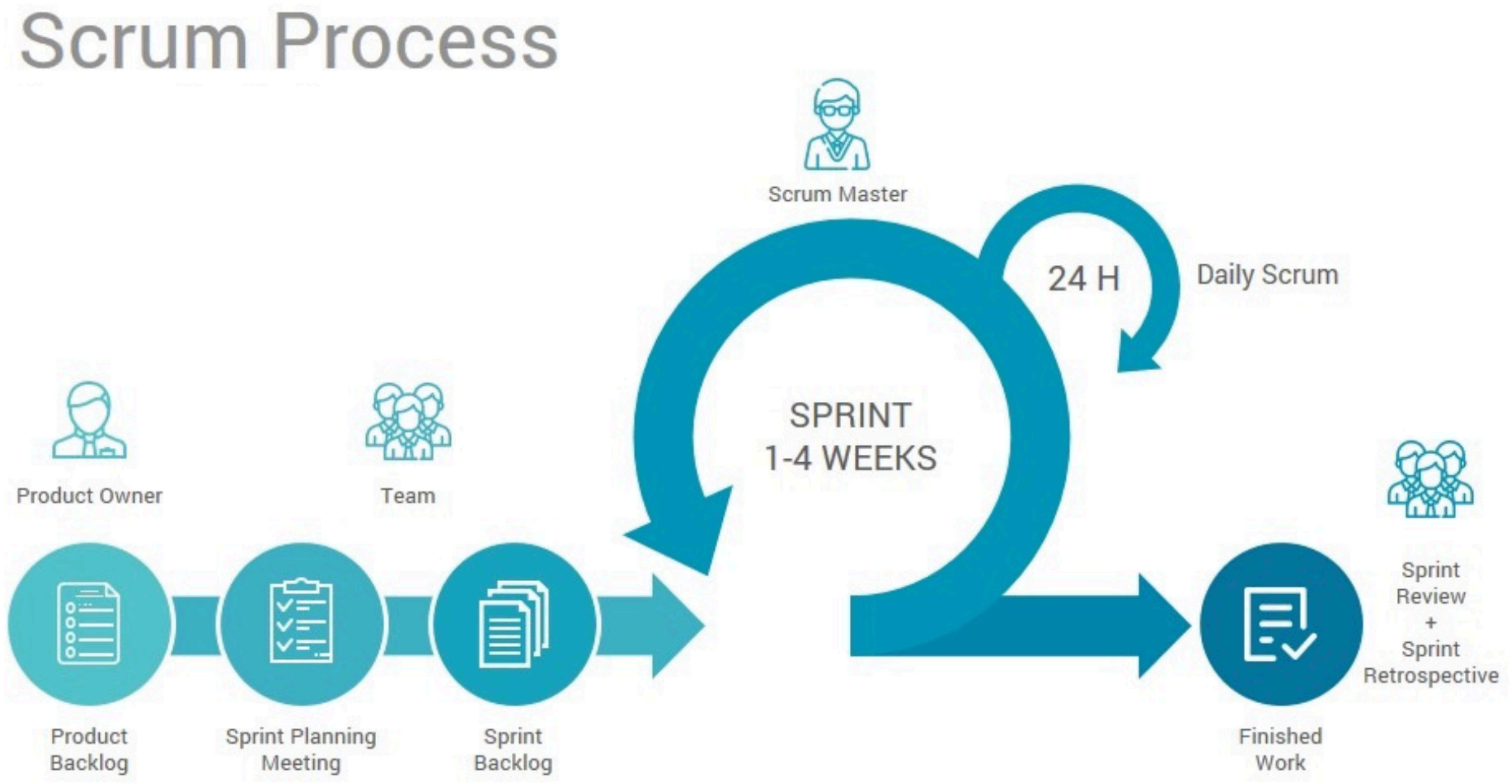
Exercise: you're an early hire and this is the situation. What do you think?

Backlogs (Product and Sprint)	~~Beha iven develo t (BDD)~~	Cross-functional team	Continuous integration (CI)	Domain-driven design (DDD)	Information radiators (Kanban board, Task board, Burndown chart)	Acceptance test-driven development (ATDD)
Iterative and incremental development (IID)	~~Pair r amming~~	~~Plan poker~~	Refactoring	Scrum meetings (Sprint planning, Daily scrum, Sprint review and retrospective)	Small releases	Simple design
Test-driven development (TDD)	Agile testing	Timeboxing	Use case	User story	~~Story-driv nodeling~~	~~Re ective~~
~~On-s ustomer~~	Agile Modeling	40-hour weeks	Short development cycles	Collective ownership	~~Oper kspace~~	~~Velo tracking~~

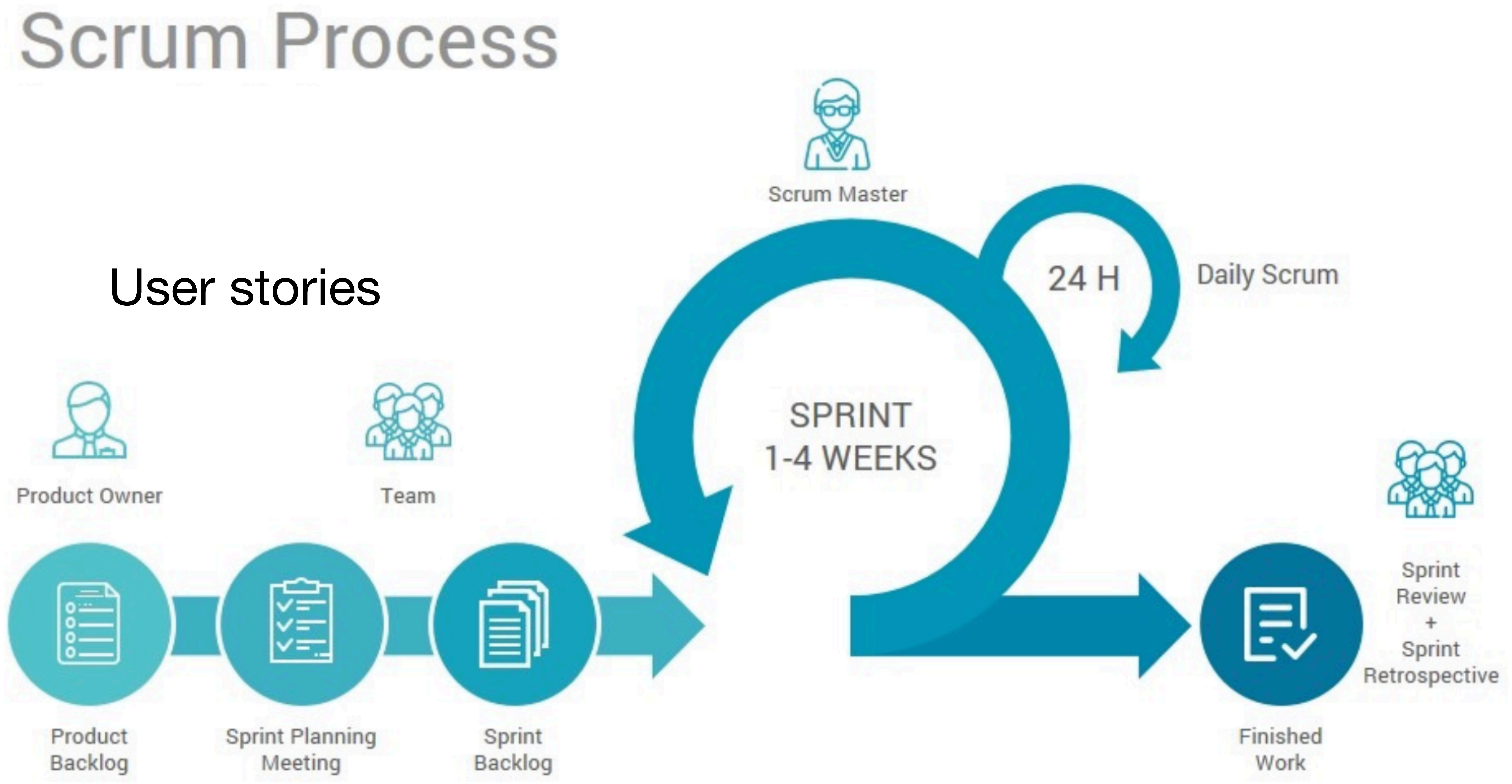
Let’s think about what’s important (Remember, what do we care most about?)

Backlogs (Product and Sprint)	Behavior-driven development (BDD)	Cross-functional team	Continuous integration (CI)	Domain-driven design (DDD)	Information radiators (Kanban board, Task board, Burndown chart)	Acceptance test-driven development (ATDD)
Iterative and incremental development (IID)	Pair programming	Planning poker	Refactoring	Scrum meetings (Sprint planning, Daily scrum, Sprint review and retrospective)	Small releases	Simple design
Test-driven development (TDD)	Agile testing	Timeboxing	Use case	User story	Story-driven modeling	Retrospective
On-site customer	Agile Modeling	40-hour weeks	Short development cycles	Collective ownership	Open workspace	Velocity tracking

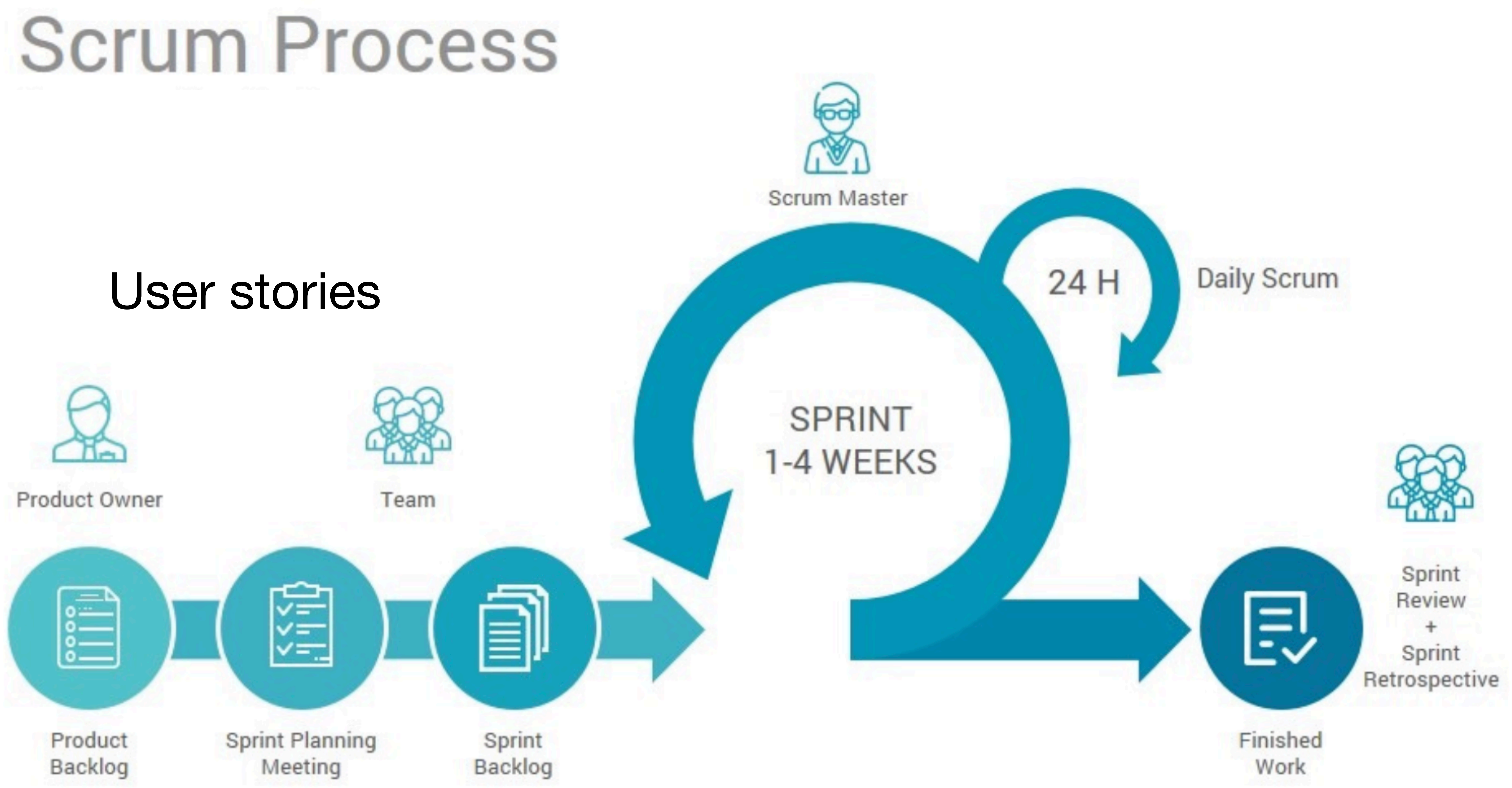
A process


Scrum Process
Scrum Master
24 H
Daily Scrum
SPRINT
1-4 WEEKS
Product Owner
Team
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work
Sprint Review + Sprint Retrospective

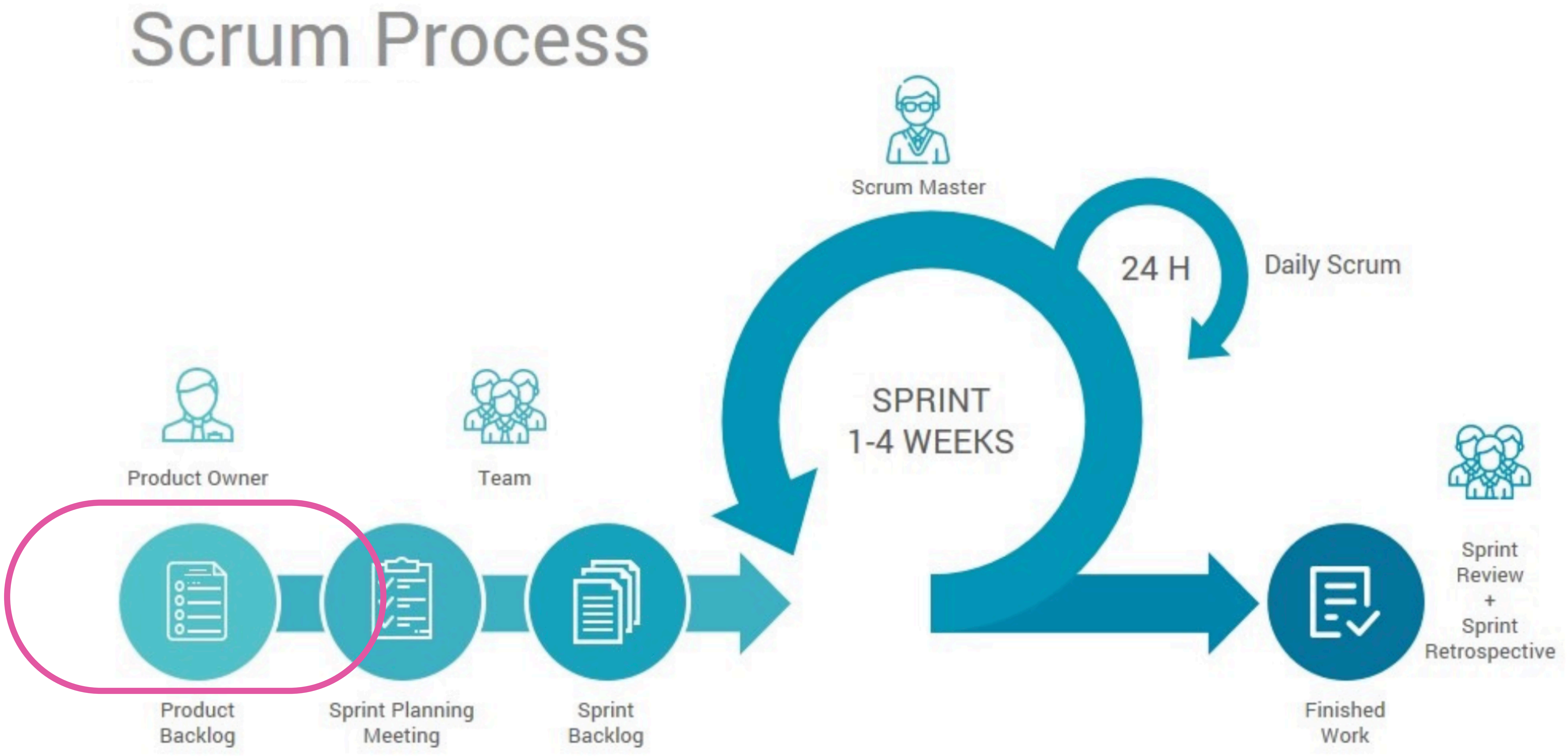
A process


Scrum Process
User stories
Scrum Master
24 H
Daily Scrum
SPRINT
1-4 WEEKS
Product Owner
Team
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work
Sprint Review
+
Sprint Retrospective

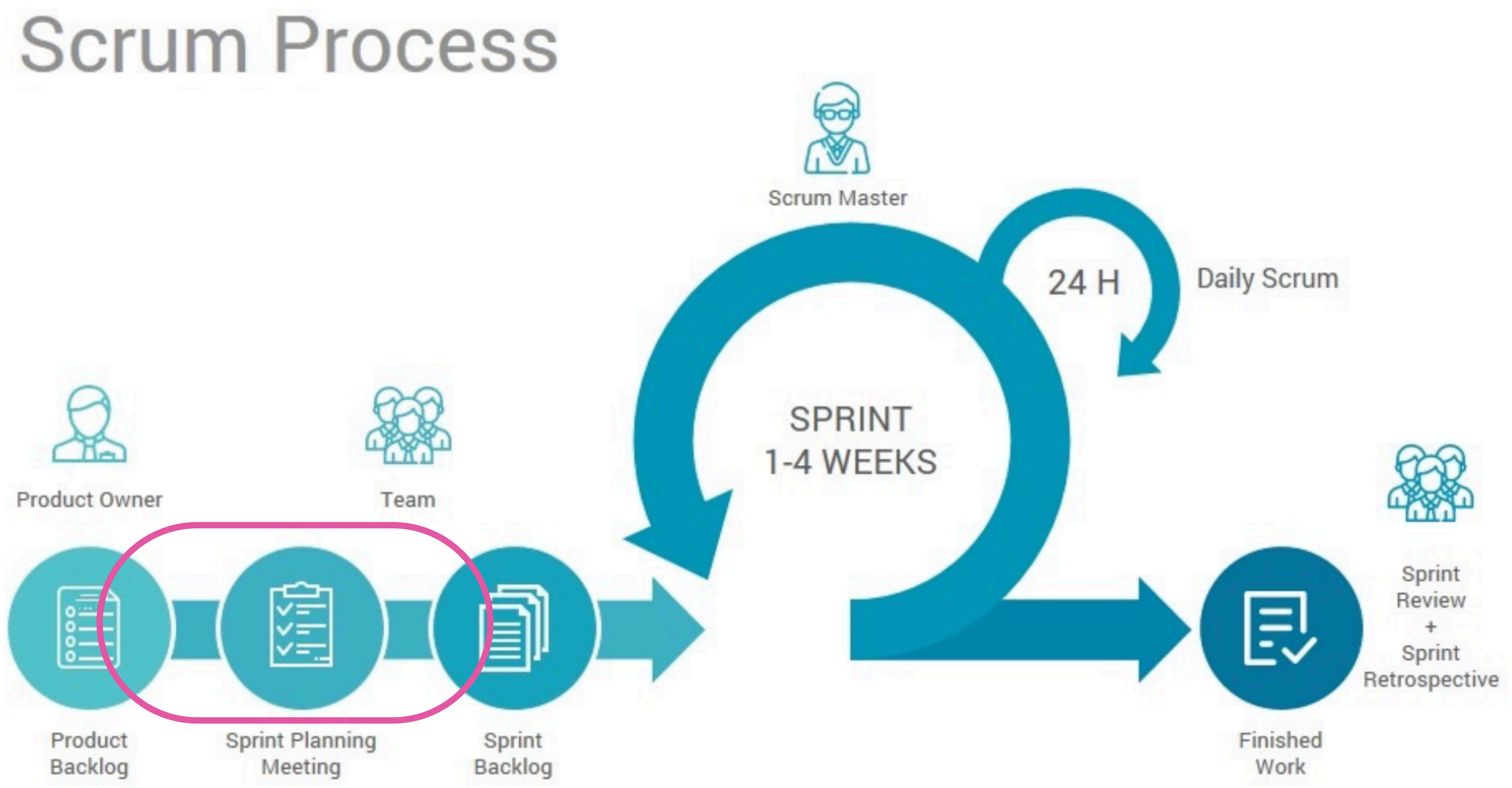
Why do we care about user stories again?



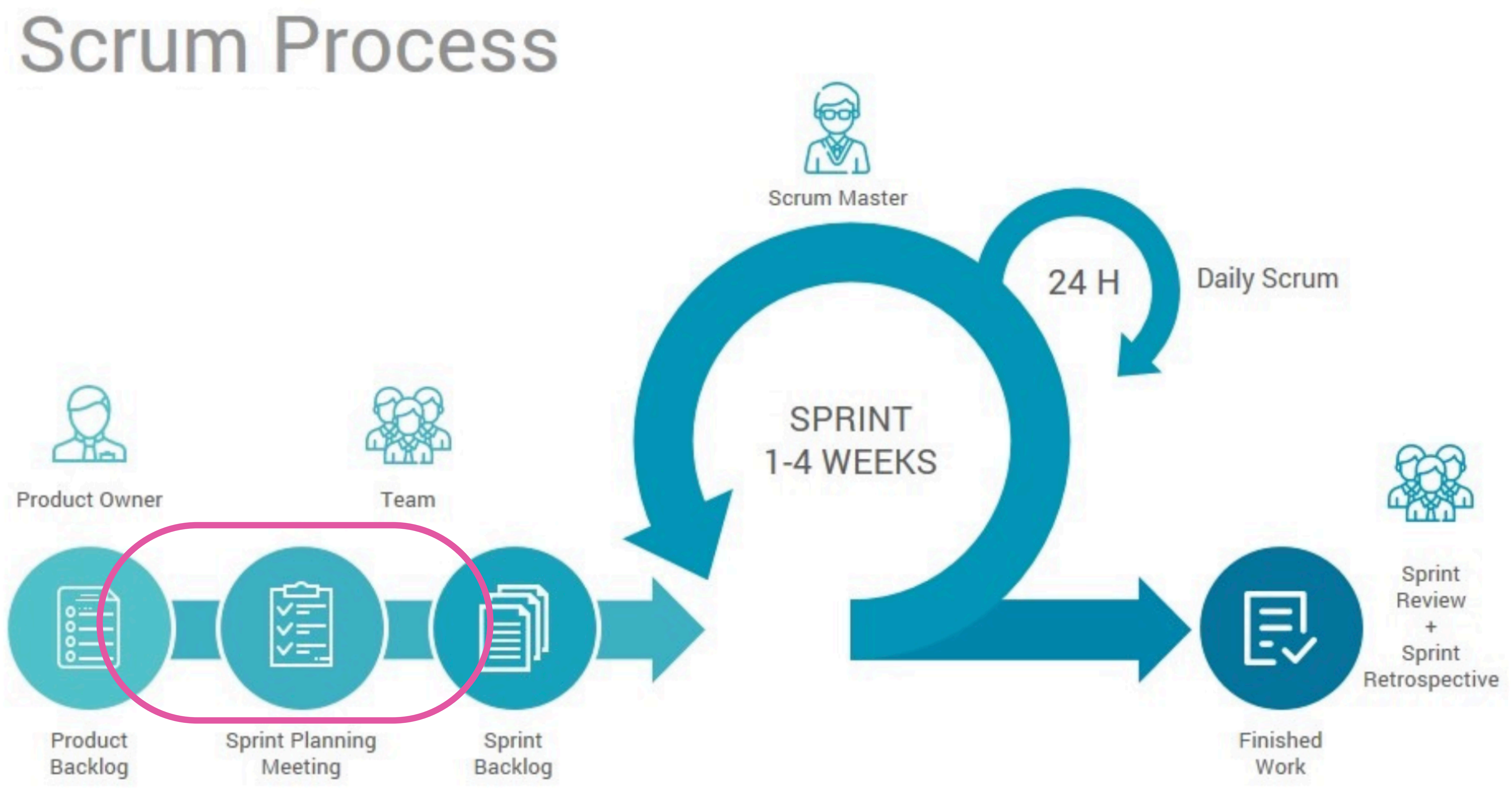
Product backlog



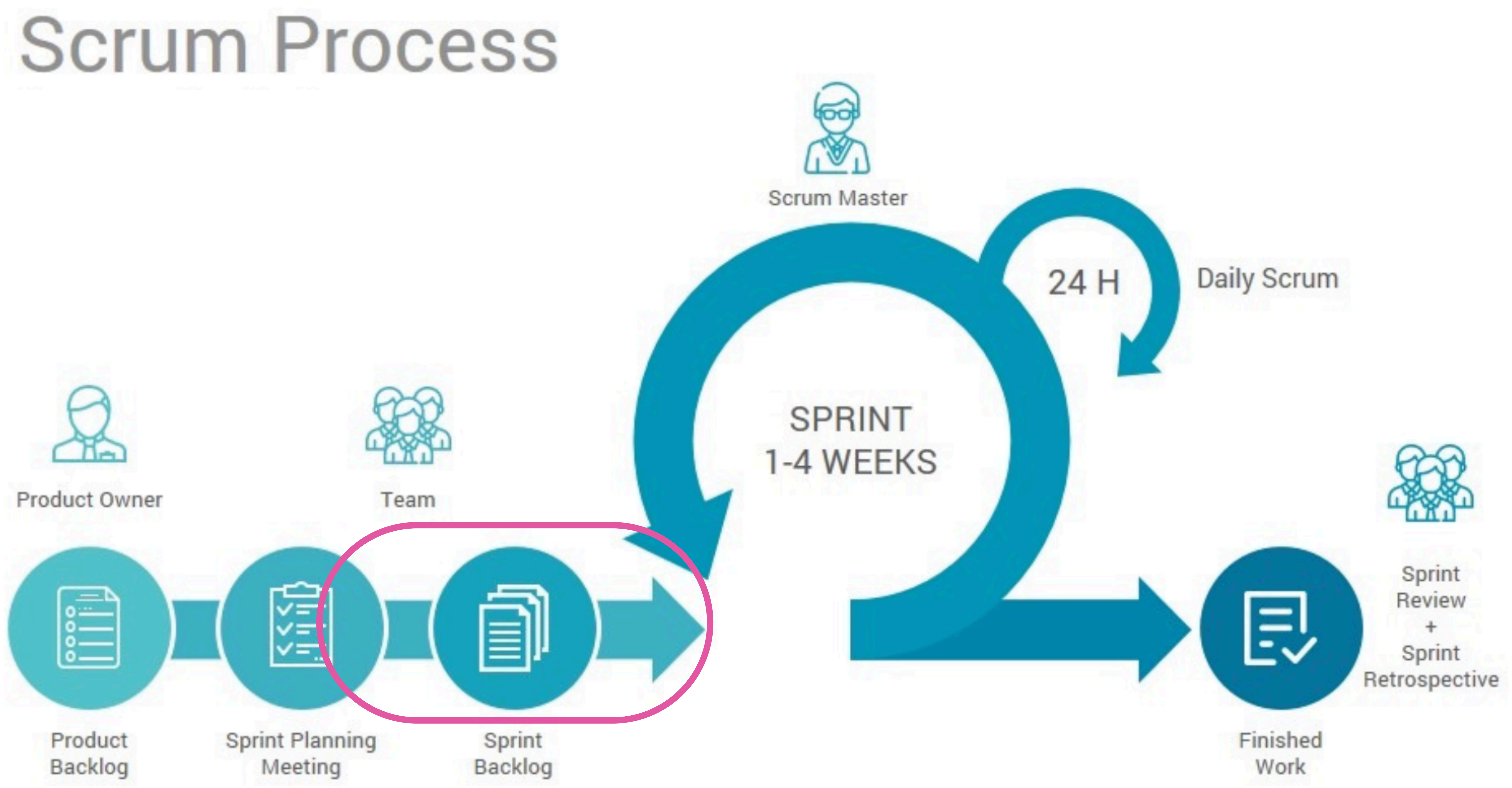
Sprint planning



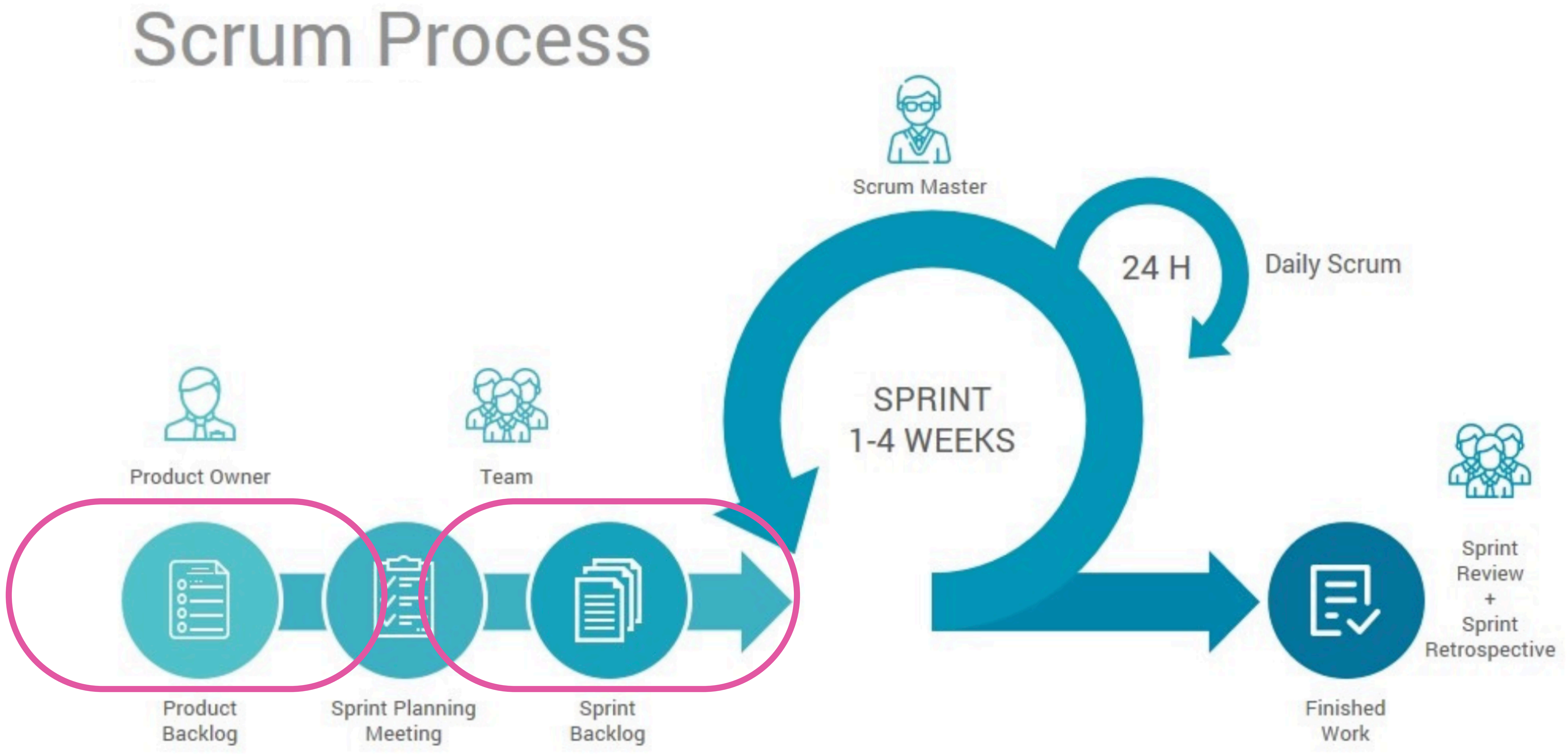
How long would it take you to do X?



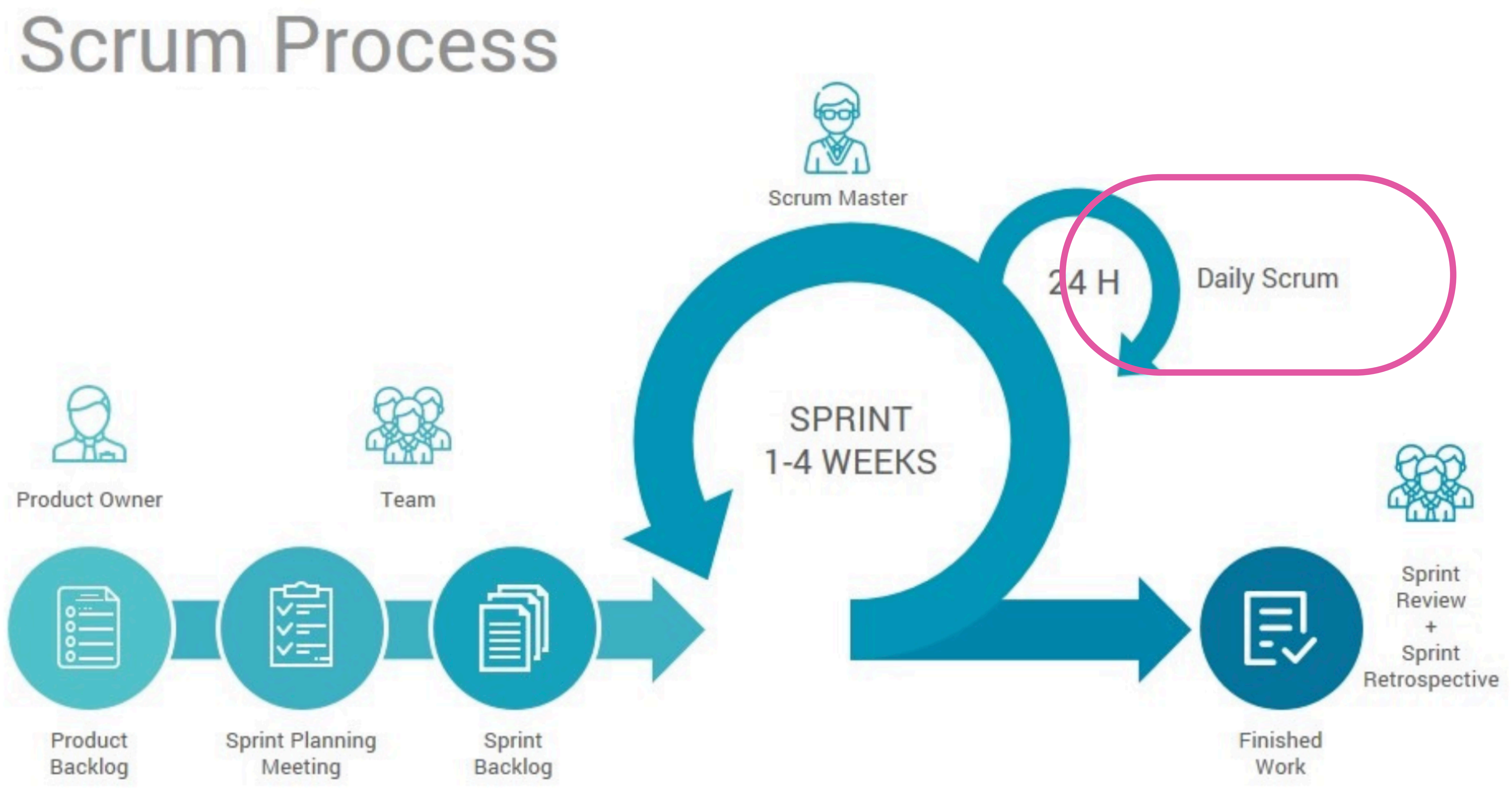
Sprint backlog



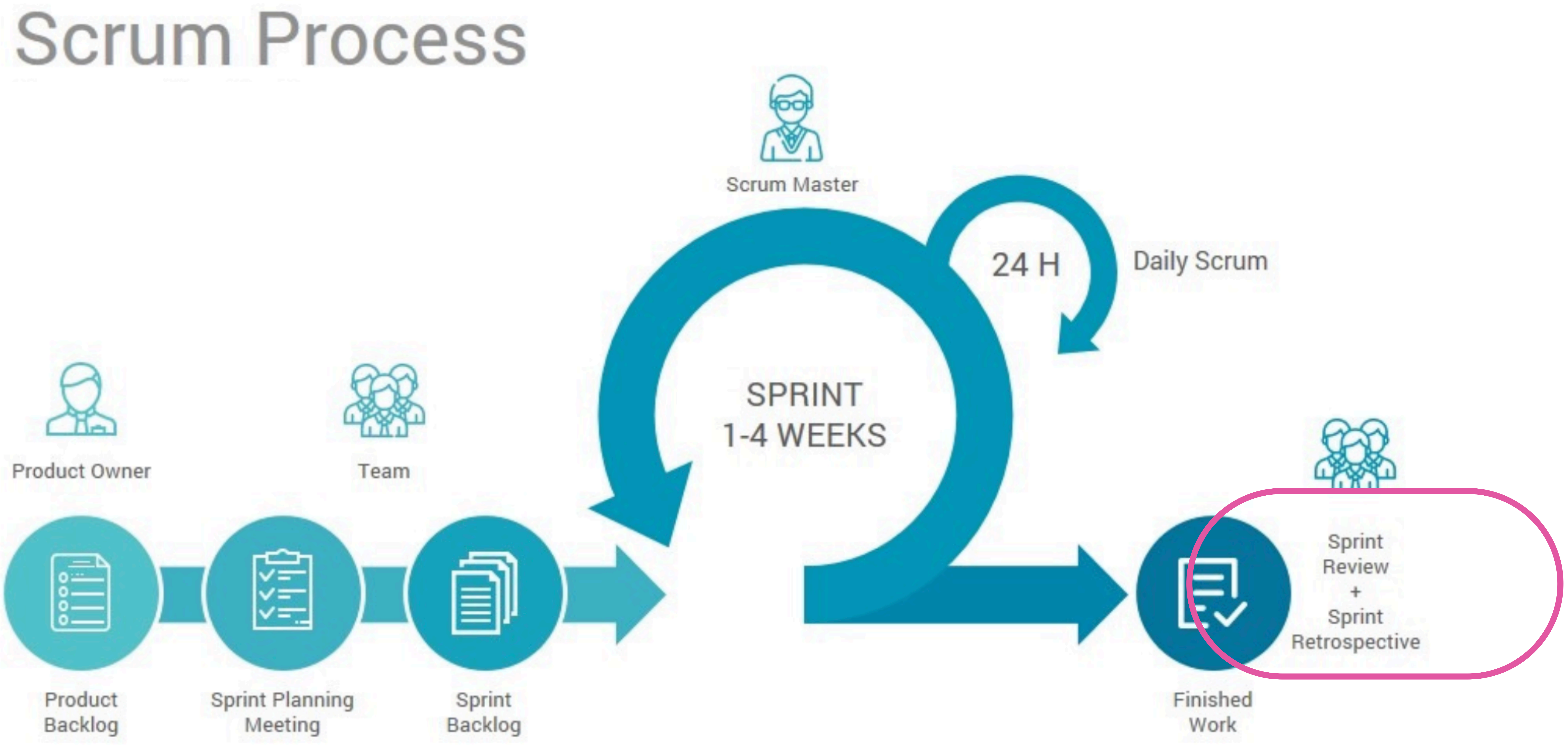
How are these different?



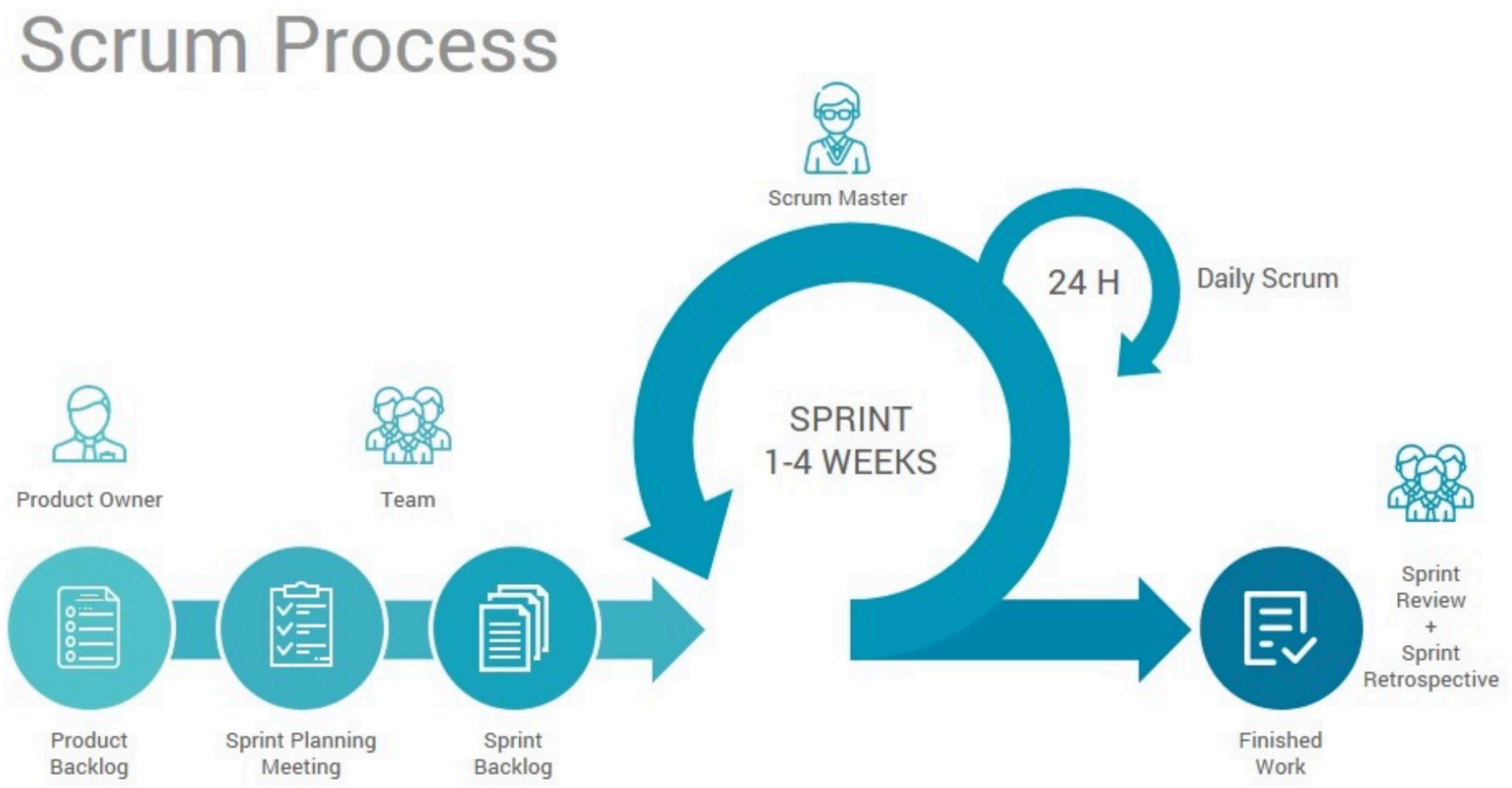
Daily scrum


Scrum Process
Scrum Master
24 H
Daily Scrum
SPRINT
1-4 WEEKS
Product Owner
Team
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work
Sprint Review + Sprint Retrospective

Sprint review and retrospective



What could we do instead?


Scrum Process
Scrum Master
24 H
Daily Scrum
SPRINT
1-4 WEEKS
Product Owner
Team
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work
Sprint Review + Sprint Retrospective