Scrum: Sprint Rules

Lecturer: Adel Vahdati

Sprints: General Rules

- A sprint spans: Sprint Planning, Sprint Execution, Sprint Review, and Sprint Retrospective.
- The following **rules** apply:
  - Sprints are time-boxed: They have fixed start and end dates.
  - Sprints are short in duration: Between one week and a calendar month.
  - Sprints are consistent in length; exceptions are only permitted under certain circumstances.
  - No goal-altering changes in scope or personnel are permitted during a sprint.
  - During each sprint, a potentially shippable product increment is completed in conformance with the Scrum team's agreed-upon "definition of done".

Sprint Rule #1: Timeboxing

- Time-boxing: A time-management technique that helps organize the performance of work and manage scope.
- Each sprint takes place in a time frame with specific start and end dates, called a **timebox**.
  - Inside this timebox, the team is expected to work at a
- **sustainable pace** to complete a chosen set of work that aligns with a sprint goal.

Timeboxing: Benefits

- **Establishes a WIP (Work In Process) Limit**
  - Because the team will plan to work on only those items that it believes it can start and finish within the sprint, timeboxing establishes a WIP limit.
- **Forces Prioritization**
  - We are forced to focus on the small amount of work that matters most.
- **Demonstrates Progress**
  - Timeboxing helps us demonstrate relevant progress by completing and validating important pieces of work by a known date (the end of the sprint).
- **Avoids Unnecessary Perfectionism**
  - Timeboxing forces an end to potentially unbounded work by establishing a fixed end date for the sprint by which a good solution must be done.
- **Motivates Closure**
  - The fact that the end of the sprint brings with it a hard deadline encourages team members to diligently apply themselves to complete the work on time.
- **Improves Predictability**
  - We can predict the work we can complete in the next short sprint.

Sprint Rule #2: Short Duration - Benefits

- **Ease of Planning**
  - It is easier to plan a few weeks' worth of work; also, planning requires far less effort and is far more accurate than longer-horizon planning.
- **Fast Feedback**
  - During each short sprint we create working software and then have the opportunity to inspect and adapt what we built and how we built it.
- **Improved Return on Investment**
  - Short-duration sprints allow for early and more frequent deliverables.
- **Bounded Error**
  - Even if we fumble the whole thing, we have lost only two weeks.
- **Rejuvenated Excitement**
  - The longer we have to wait for gratification, the faster our interest will decline; short-duration sprints keep participant excitement high.
- **Frequent Checkpoints**
  - At the end of each short sprint there is a checkpoint (the sprint review) that allows everyone to base decisions on demonstrable, working features.

Sprint Rule #3: Consistent Duration

- On a development effort, a team should pick a consistent duration for its sprints and not change it unless there is a compelling reason.
- Compelling reasons might include the following:
  - You want to try a couple of trial sprints before making a final decision on the sprint duration.
  - Public holidays make it more practical to change the duration.
  - Product release occurs in two weeks, so a longer sprint would be wasteful.
- Bad reason:
  - The team cannot get all the work done within the current sprint length.
- A week usually means five calendar weekdays.
  - If there is a one-day holiday or training event during the sprint, it reduces the team's capacity for that sprint but does not necessitate a length change.

Consistent Duration: Benefits

- **Cadence** (a regular, predictable rhythm or heartbeat)
  - It allows us to acquire a rhythmic familiarity with when things need to happen to achieve the fast, flexible flow of business value.
  - It enables people to get comfortable with the project.
  - It tends to level out the intensity of work: We do not see a steep increase in intensity in the latter phases, so teams can work at a sustainable pace.
  - It significantly reduces coordination overhead: We can predictably schedule sprint activities for many sprints at the same time.
  - If we have multiple teams on the same project, cadence allows for synchronization of the work across all of the teams.
- **Simplified Planning**
  - Velocity is typically normalized to a sprint; if the length of the sprint can vary, we will not have a normalized sprint unit.
  - If the length of the sprint can vary, calculating the number of sprints in the release could be challenging and involve unnecessary overhead.

Sprint Rule #4: No Goal-Altering Changes

- Once the sprint goal has been established and sprint execution has begun, no change is permitted that can alter the sprint goal.
- A sprint goal describes the business purpose and value of the sprint. It typically has a clear, single focus; some examples are given below:
  - Support initial report generation.
  - Load and curate North America map data.
  - Get basic printing working and support search by date.
- The sprint goal is the foundation of a mutual commitment made by the team and the product owner.
  - The team commits to meeting the goal by the end of the sprint, and the product owner commits to not altering the goal during the sprint.

No Goal-Altering Changes: Change vs. Clarification

- Although the sprint goal should not be materially changed, it is permissible to clarify the goal.
- What constitutes a change?
  - A change is any alteration in work or resources that
    - has the potential to generate economically meaningful waste, or
    - harmfully disrupt the flow of work, or
    - substantially increase the scope of work within a sprint.
- Examples of goal change:
  - Adding or removing a product backlog item from a sprint.
  - Altering the scope of a product backlog item that is already in the sprint.
- What constitutes a clarification?
  - Clarifications are additional details provided during the sprint that assist the team in achieving the sprint goal.

No Goal-Altering Changes: Consequences of Change

- Change has consequences: We have to embrace change in a balanced, economically sensible way.
  - Once a sprint starts, our investment in its PBIs increases.
    - If we make a change after sprint planning, we not only jeopardize the planning investment, but we also incur additional costs for having to re-plan.
    - Investment in work increases as PBIs transition through the states of to do (work not yet started), doing (work in process), and done (work completed).
  - The economics can be indirectly affected by the potential deterioration of team motivation and trust that can accompany a change.
- The no-goal-altering-change rule is not a law. The Scrum team has to be pragmatic:
  - If the economic consequences of the change are far less than the economic consequences of deferring the change, apply the change.
  - If the sprint goal becomes invalid, the Scrum team may decide that continuing the sprint makes no sense and advise the product owner to terminate it.

Sprint Rule #5: Conformance to the Definition of Done

- Definition of Done: A checklist of the types of work that the team is expected to complete before it can declare its work as potentially shippable.
- The items on the checklist will depend on a number of variables:
  - The nature of the product being built
  - The technologies being used to build it
  - The organization that is building it
  - The current impediments that affect what is possible

	Definition of Done
❑	Design reviewed
❑	Code completed
❑	Code refactored
❑	Code in standard format
❑	Code is commented
❑	Code checked in
❑	Code inspected
❑	End-user documentation updated
❑	Tested
❑	Unit tested
❑	Integration tested
❑	Regression tested
❑	Platform tested
❑	Language tested
❑	Zero known defects
❑	Acceptance tested
❑	Live on production servers

Definition of Done: Evolvability

- The Definition of Done can evolve over time:
  - Many teams, start out with a definition of done that does not end in a state where all features are potentially shippable.
    - Real organizational impediments might prevent them from reaching this state at the start of development, even though it is the ultimate goal.
    - As a result, they might (necessarily) start with a lesser end state and let their definition of done evolve over time as impediments are removed.
- Example:
  - Products that include both hardware and software, where hardware is late.
    - The software team will not have the hardware on which to test the software, so it cannot really claim that the results produced are potentially shippable.
    - At first it might claim "emulator done," as testing during the early sprints is typically performed against a software emulator of the actual hardware.

Definition of Done: Difference with Acceptance Criteria

- The definition of done applies to the product increment being developed during the sprint.
  - The product increment consists of a set of product backlog items, so each backlog item must be completed in conformance with the definition-of-done checklist.
- Each product backlog item should also have a set of conditions of satisfaction (item-specific acceptance criteria), specified by the product owner.
  - These acceptance criteria eventually will be verified in acceptance tests that the product owner will confirm to determine if the backlog item functions as desired.
  - E.g., if the product backlog item is "Allow a customer to use a credit card," the conditions of satisfaction might be "Works with AmEx, Visa, and MasterCard."
- These item-specific criteria are in addition to, not instead of, the criteria specified by the definition-of-done checklist.
  - A product backlog item can be considered done only when both the item-specific acceptance criteria and the sprint-level definition-of-done criteria have been met.

References

- Rubin, K.S., Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.
- Schwaber, K., Sutherland, J., The Scrum Guide, Published online at: http://www.scrumguides.org/, July 2013 (last visited on: 7 November 2014).
- Ramsin, Raman. "Home." Department of Computer Science and Engineering, Sharif University of Technology. Accessed February 15, 2025. https://sharif.edu/~ramsin/index.htm.