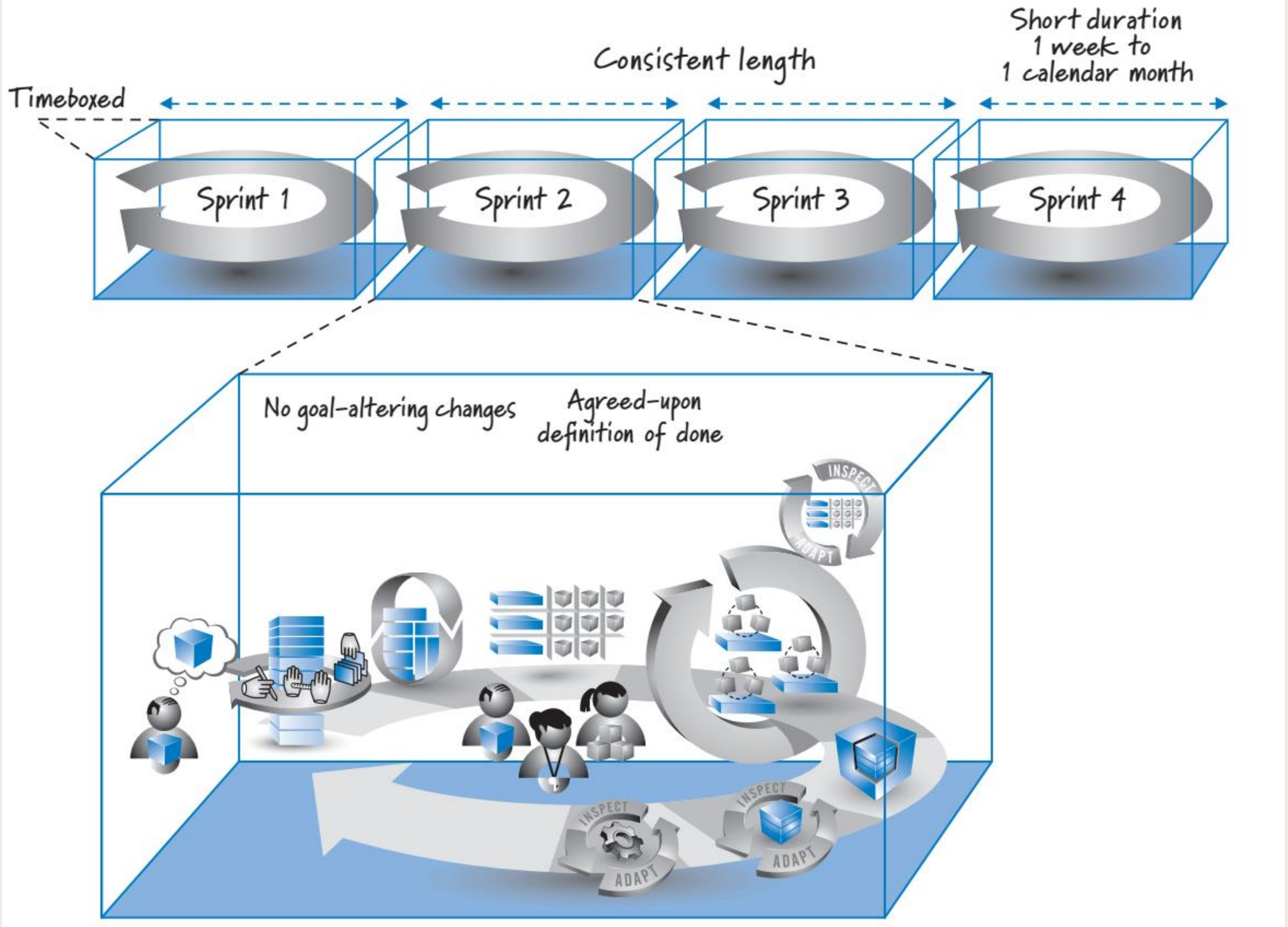
Sprint Rules(II)

Dr. Elham Mahmoudzadeh
Isfahan University of Technology
mahmoudzadeh@iut.ac.ir
2023

Introduction

- Scrum organizes work in iterations or cycles of up to a calendar month called sprints.
- Sprints are the skeleton of the Scrum framework.
- A sprint spans: Sprint Planning, Sprint Execution, Sprint Review, and Sprint Retrospective.


Short duration
1 week to
1 calendar month
Consistent length
Timeboxed
Sprint 1
Sprint 2
Sprint 3
Sprint 4
No goal-altering changes
Agreed-upon
definition of done
INSPECT
ADAPT
INSPECT
ADAPT
INSPECT
ADAPT

Sprint Rules

- All sprints are timeboxed: They have fixed start and end dates.
- Sprints must also be short: Between one week and a calendar month.
- Sprints should be consistent in length, though exceptions are 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.”

Third Rule: **Consistent Duration**(I)

- As a rule, on a given development effort, a team should pick a consistent duration for its sprints and not change it unless there is a compelling reason.
  - You are considering moving from four-week sprints to two-week sprints in order to obtain more frequent feedback but want to try a couple of two-week sprints before making a final decision.
  - The annual holidays or end of the fiscal year make it more practical to run a three-week sprint than the usual two-week sprint.
  - The product release occurs in one week, so a two-week sprint would be wasteful.

Third Rule: Consistent Duration(II)

- The fact that the team cannot get all the work done within the current sprint length is not a compelling reason to extend the sprint length.
- Neither is it permissible to get to the last day of the sprint, realize you are not going to be done, and lobby for an extra day or week. These are symptoms of dysfunction; they are not good reasons to change the sprint length.

Third Rule: **Consistent Duration**(III)

- 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 doesn’t necessitate a sprint length change.

Cadence as a benefit of Consistent Duration(I)

- Sprints of the same duration provide us with cadence: regular, predictable rhythm or heartbeat to a Scrum development effort.
- A steady, healthy heartbeat allows the Scrum team and the organization to acquire an important rhythmic familiarity with when things need to happen to achieve the fast, flexible flow of business value.

Cadence as a benefit of Consistent Duration(II)

- Having a regular cadence to sprints enables people to “get into the zone,”
- This happens because regular cadence makes the mundane but necessary activities habitual, thereby freeing up mental capacity to stay focused on the fun, value-added work.
- Generally, It enables people to get comfortable with the project.

Cadence as a benefit of Consistent Duration(III)

- Having a short sprint cadence also tends to level out the intensity of work.
- Unlike a traditional sequential project where we see a steep increase in intensity in the latter phases, each sprint has an intensity profile that is similar to that of the other sprints.
- Sprint cadence enables teams to work at a sustainable pace.

Cadence as a benefit of Consistent Duration(IV)

- Sprinting on a regular cadence also significantly reduces coordination overhead.
- With fixed-length sprints we can predictably schedule the sprint-planning, sprint review, and sprint retrospective activities for many sprints at the same time.
- Because everyone knows when the activities will occur, the overhead required to schedule them for a large batch of sprints is substantially reduced.
- If we allowed sprint durations to vary from sprint to sprint, imagine the extra effort we would need to coordinate the schedules of the stakeholders on what might be just one or two weeks' notice for an upcoming sprint review!

Cadence as a benefit of Consistent Duration(V)

- If we have multiple teams on the same project, having all teams with a similar sprint cadence allows for synchronization of the work across all of the teams.

Simplified planning as a benefit of Consistent Duration(I)

- When all sprints are the same length (even when they might have a day or less capacity per sprint because of a holiday), the team gets comfortable with the amount of work that it can accomplish in a typical sprint (referred to as its velocity).
- Velocity is typically normalized to a sprint. If the length of the sprint can vary, we really don’t have a normalized sprint unit.
- While it is certainly possible to compute a team’s velocity even if it uses variable length sprints, it is more complicated.
- Sticking with a consistent sprint duration simplifies the computations we perform on a team’s historical velocity data.

Simplified planning as a benefit of Consistent Duration(II)

- Consistent sprint durations also simplify the rest of the planning.
- If the sprint durations were allowed to vary, calculating the number of sprints in the release could be significantly more challenging (because we would have to do extensive early planning), involve unnecessary overhead, and likely be far less reliable than with consistent sprint durations.

Forth Rule: No Goal-Altering Changes(I)

- Once the sprint goal has been established and sprint execution has begun, no change is permitted that can materially affect the sprint goal.

What Is a Sprint Goal?

- Each sprint can be summarized by a sprint goal that describes the business purpose and value of the sprint. Typically the sprint goal has a clear, single focus.
- Examples
  - Support initial report generation.
  - Load North America map data.
- During sprint planning, the development team should help refine and agree to the sprint goal and use it to determine the product backlog items that it can complete by the end of the sprint.

No Goal-Altering Changes: **Mutual Commitment**

- The sprint goal is the foundation of a mutual commitment made by the team and the product owner.
- The **team commits to meet the goal by the end of the sprint**, and the **product owner commits to not altering the goal during the sprint**.
- This mutual commitment demonstrates the importance of sprints in balancing the needs of the business to be adaptive to change, while allowing the team to concentrate and efficiently apply its talent to create value during a short, fixed duration.
- By defining and adhering to a sprint goal, the Scrum team is able to stay focused (in the zone) on a well-defined, valuable target.

No Goal-Altering Changes: **Change versus Clarification**(I)

- Although the sprint goal should not be materially *changed*, it is permissible to *clarify* the goal.
- A **change** is any alteration in work or resources that has the potential to generate economically meaningful waste, harmfully disrupt the flow of work, or substantially increase the scope of work within a sprint. Adding or removing a product backlog item from a sprint or significantly altering the scope of a product backlog item that is already in the sprint typically constitutes change.
- Example of goal changes: Adding the ability to search based on a picture likely represents substantially more effort and almost certainly would affect the team’s ability to meet a commitment to deliver search based on last name and first name.

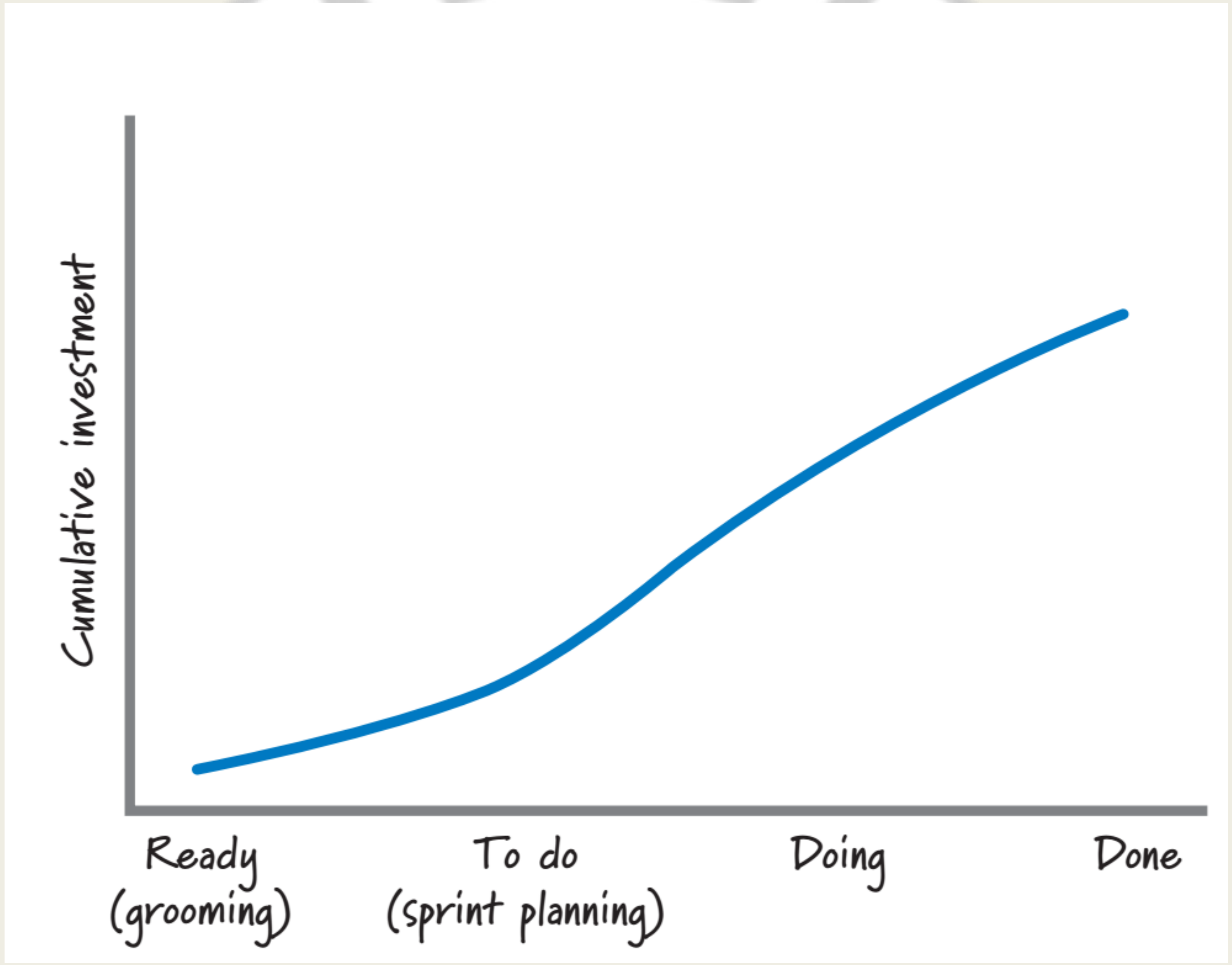
No Goal-Altering Changes: **Change versus Clarification**(II)

- **Clarifications** are additional details provided during the sprint that assist the team in achieving its sprint goal.
- All of the details associated with product backlog items might not be fully known or specified at the start of the sprint. Therefore, it is completely reasonable for the team to ask clarifying questions during a sprint and for the product owner to answer those questions.
- Example of clarification: did you have a preference for how that list is to be ordered? Yes, order alphabetically.

No Goal-Altering Changes: **Consequences of Change**(I)

- It may appear that the no-goal-altering-change rule is in direct conflict with the core Scrum principle that we should embrace change. We do embrace change, but we want to embrace it in a balanced, economically sensible way.
- The economic consequences of a change increase as our level of investment in the changed work increases.

Cumulative investment at different states



No Goal-Altering Changes: **Consequences of Change**(III)

- We invest in product backlog items to get them ready to be worked on in a sprint.
- However, once a sprint starts, our investment in those product backlog items has increased (because we spent time during sprint planning to discuss and plan them at a task level).
- If we want to make a change after sprint planning has occurred, we not only jeopardize the planning investment, but we also incur additional costs for having to replan any changes during the sprint.

No Goal-Altering Changes: **Consequences of Change**(IV)

- In addition, once we begin sprint execution, our investment in work increases even more as product backlog items transition through the states of to do (work not yet started), doing (work in process), and done (work completed).

No Goal-Altering Changes: **Consequences of Change**(V)

- Let's say we want to swap out feature X, currently part of the sprint commitment, and substitute feature Y, which isn't part of the existing commitment.
- Even if we haven't started working on feature X, we still incur planning waste. In addition, feature X might also have dependencies with other features in the sprint, so a change that affects feature X could affect one or more other features, thus amplifying the effect on the sprint goal.

No Goal-Altering Changes: **Consequences of Change**(VI)

- If work on feature X has already begun, in addition to the already-mentioned waste, we could have other potential wastes.
- For example, all of the work already performed on feature X might have to be thrown away. And we might have the additional waste of removing the partially completed work on feature X, which we may never use in the future.
- And, of course, if feature X is already completed, we might have wasted the full investment we made in feature X. All of this waste adds up!

No Goal-Altering Changes: **Consequences of Change**(VII)

- In addition to the direct economic consequences of waste, the economics can be indirectly affected by the potential deterioration of team motivation and trust that can accompany a change.
- When the product owner makes a commitment to not alter the goal and then violates the commitment, the team naturally will be demotivated, which will almost certainly affect its desire to work diligently to complete other product backlog items.
- In addition, violating the commitment can harm the trust within the Scrum team, because the development team will not trust that the product owner is willing to stick to his commitments.

Being Pragmatic(I)

- The no-goal-altering-change rule is just that—a rule, not a law. The Scrum team has to be pragmatic.
- What if business conditions change in such a way that making a change to the sprint goal seems warranted? Say a competitor launches its new product during our sprint. After reviewing the new product, we conclude that we need to alter the goal we established for our current sprint because what we are doing is now economically far less valuable given what our competitor has done. Should we blindly follow the rule of no goal-altering changes and not alter our sprint? Probably not.
- What if a critical production system has failed miserably and some or all of the people on our team are the only ones who can fix it? Should we not interrupt the current sprint to fix it? Do we tell the business that we will fix the production failure first thing next sprint? Probably not.

Being Pragmatic(II)

- In the end, being pragmatic trumps the no-goal-altering-change rule. We must act in an economically sensible way.
- If the economic consequences of the change are far less than the economic consequences of deferring the change, making the change is the smart business decision.
- If the economics of changing versus not changing are immaterial, no change to the sprint goal should be made.
- As for team motivation and trust, when a product owner has a frank, economically focused discussion with the team about the necessity of the change, most teams understand and appreciate the need, so the integrity of motivation and trust is upheld.

Abnormal Termination(I)

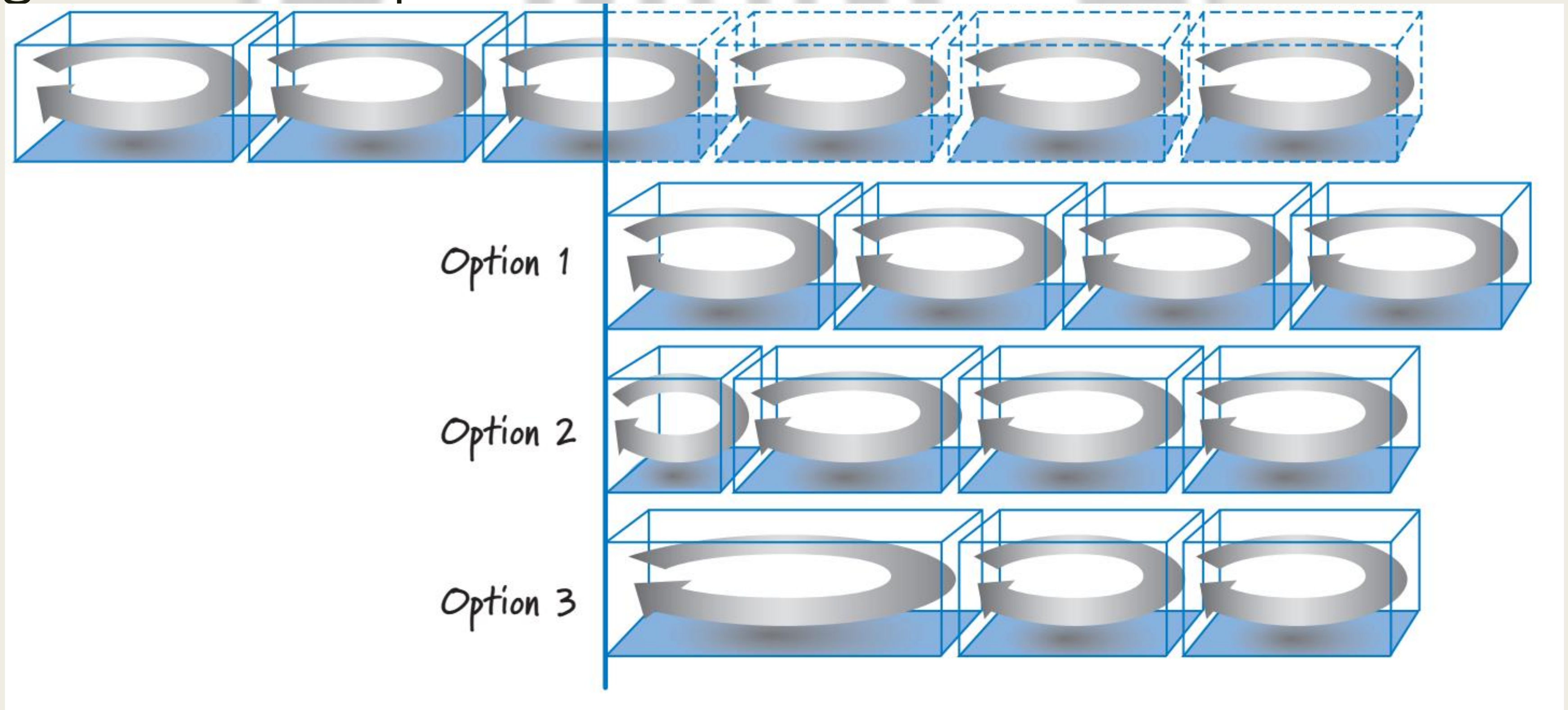
- When the sprint goal become completely invalid, the Scrum team may decide that continuing with the current sprint makes no sense and advise the product owner to abnormally terminate the sprint.
- When a sprint is abnormally terminated, the current sprint comes to an abrupt end and the Scrum team gathers to perform a sprint retrospective. The team then meets with the product owner to plan the next sprint, with a different goal and a different set of product backlog items.
- Sprint termination is used when an economically significant event has occurred, such as a competitor’s actions that completely invalidate the sprint or product funding being materially changed.

Abnormal Termination(II)

- It is important to realize that terminating the sprint early, in addition to having a negative effect on morale, is a serious disruption of the fast, flexible flow of features and negates many of the benefits of consistent-duration sprints.
- Terminating a sprint should be the last resort.

Options in the case of termination(I)

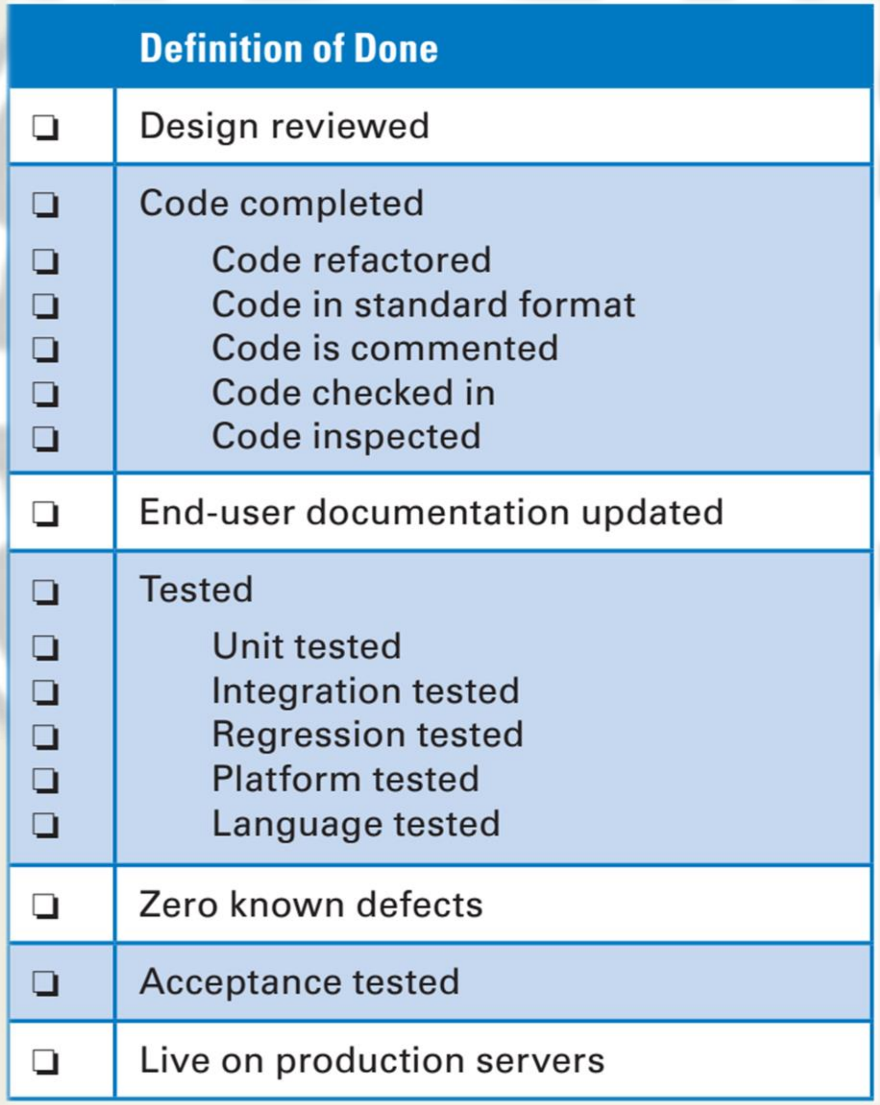
- If a sprint is terminated, the Scrum team will have to determine the length of the next sprint.



Fifth Rule: Conformance to the Definition of Done

- Definition of done is a checklist of the types of work that the team is expected to successfully complete before it can declare its work to be potentially shippable.
- Items on the checklist will depend on
  - *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.*

Example of definition of done



	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

- What if there is a significant defect that remains on the last day of the sprint; is the product backlog item done? No, it’s not done!
- And because, as a rule, we don’t extend sprints beyond the end of the planned timebox, we wouldn’t extend the sprint by a day or two to fix the defect in the current sprint.
- Instead, at the planned end of the sprint, the incomplete product backlog item is taken from the current sprint and reinserted into the product backlog in the proper order based on the other items that are currently in the product backlog. The incomplete item might then be finished in some future sprint.

Definition of Done

- Scrum teams need to have a robust definition of done, one that provides a high level of confidence that what they build is of high quality and can be shipped.
- “potentially shippable” doesn’t mean that what was built must actually be shipped. Shipping is a business decision that often occurs at a different cadence; in some organizations it may not make sense to ship at the end of every sprint.

Definition of Done Can Evolve Over Time

- For some, real impediments might prevent team from reaching defined state of the work at the end of the sprint, 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 organizational impediments are removed.

Example

- It is a product that includes hardware and software, where the hardware is late.
- If the team is building software and it doesn't have the actual hardware on which to test the software, it can't really claim that the results produces at the end of the sprint are potentially shippable.
- At best it might claim "emulator done," because testing during the early sprints is typically performed against a software emulator of the actual hardware. Later, when the actual hardware is available, the definition of done will evolve to mean potentially shippable or at least something closer to it.

Definition of Done versus Acceptance Criteria(I)

- The **definition of done** applies to the product increment being developed during the sprint. The product increment is composed of a set of product backlog items, so each backlog item must be completed in conformance with the work specified by the definition-of-done checklist.
- Each product backlog item that is brought into the sprint should have a set of **conditions of satisfaction** (item-specific acceptance criteria), specified by the product owner.

Definition of Done versus Acceptance Criteria(II)

- Acceptance criteria eventually will be verified in acceptance tests that the product owner will conform to determine if the backlog item functions as desired.
- For example, if the product backlog item is “Allow a customer to purchase with a credit card,” the conditions of satisfaction might be “Works with AmEx, Visa, and MasterCard.”
- So each product backlog item will have its own appropriate set of acceptance criteria. These item-specific criteria are the done criteria specified by the definition-of-done checklist, which apply to all product backlog items.

Done versus Done-Done

- Done: doing as much work as they are prepared to do.
- Done-Done: doing the work required for customers to believe it is done.

Reference

1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.