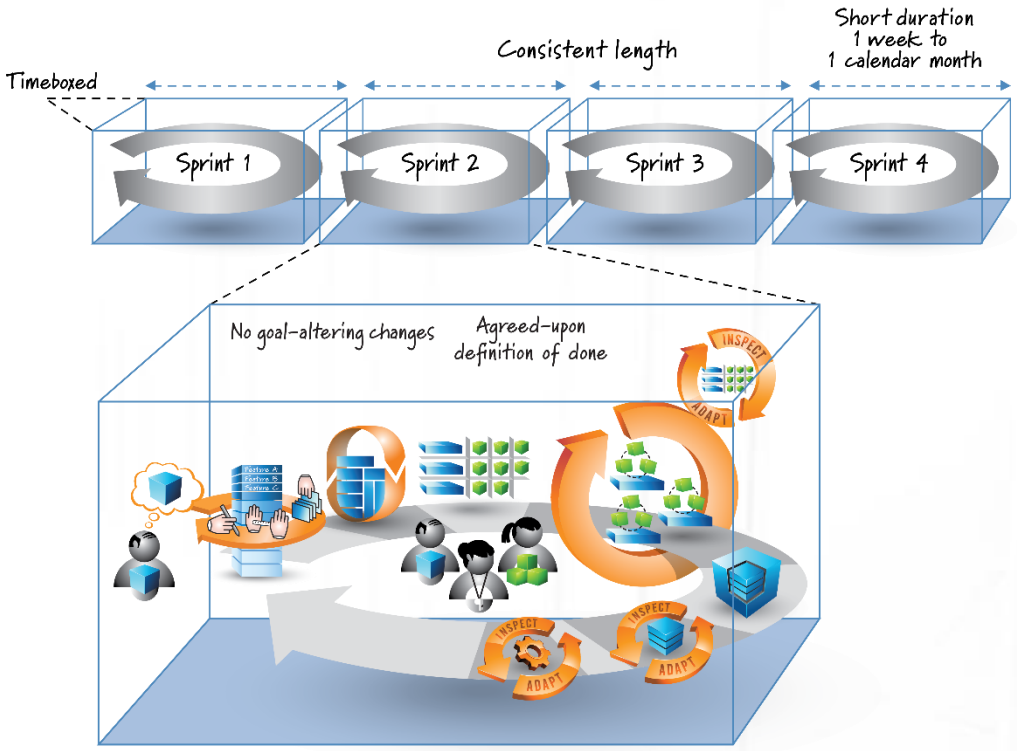
Sprints

CEN 4010 Intro to Software Engineering

Professor Alex Roque

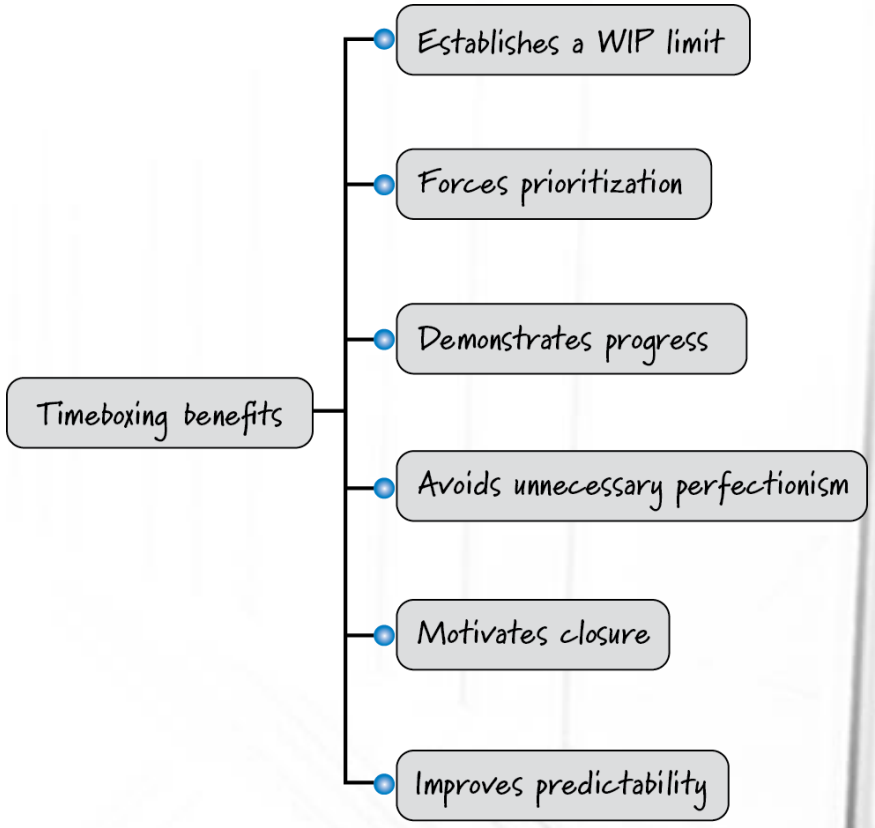
Sprints

- **Scrum** organizes work in iterations or cycles, called **Sprints**, of up to a calendar month
- **Sprint** key characteristics:
  - **Timeboxed**
  - **Short** and **consistent** duration
  - **Goal** should not be altered once started
  - Must reach the **end-state** specified by the team's definition of **done**



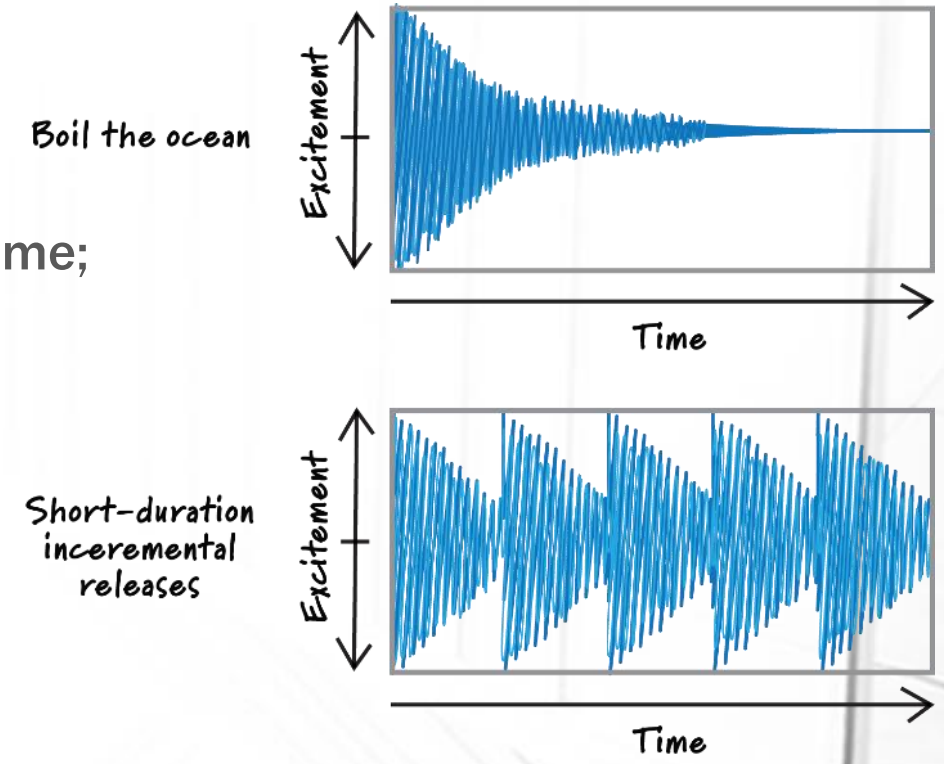
Timeboxing

- **Time-management** technique that helps **organize** the **performance** of work and manage its **scope**
  - Establishes a Work-in-Process (WIP) limit for the team to both start & finish
  - Forces Prioritization
  - Demonstrates Progress
  - Avoids Unnecessary perfection – “good enough” often suffices
  - Motivates Closure due to known short, end date
  - Improves Predictability of short-term work being done



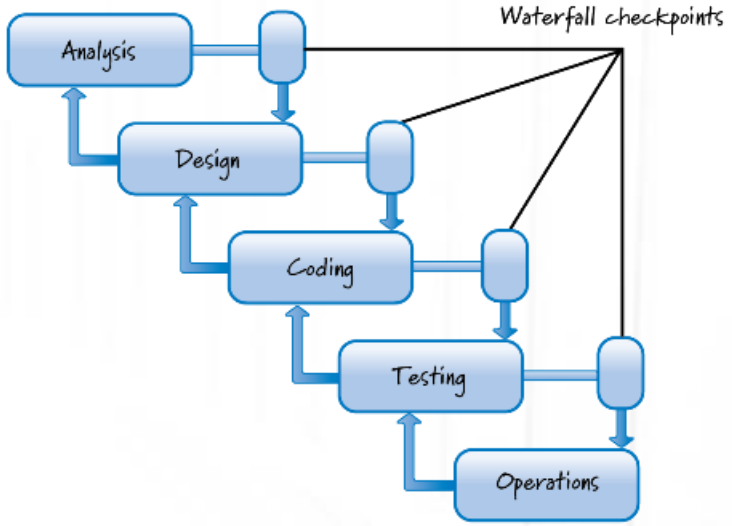
Why Short Duration is beneficial!

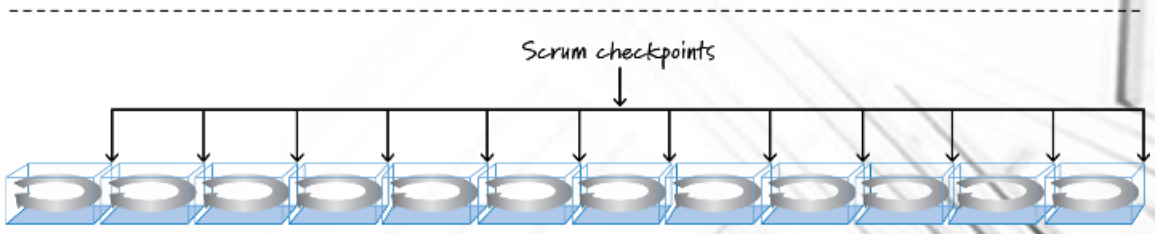
- Easier to Plan
- Fast Feedback
- Bounded Error – may only lose short amount of time; provides for frequent coordination and feedback
- Improved Return on Investment – may be able to generate revenue sooner
- Rejuvenated Excitement – short-term success/gratification
- Frequent Checkpoints



What about checkpoints?

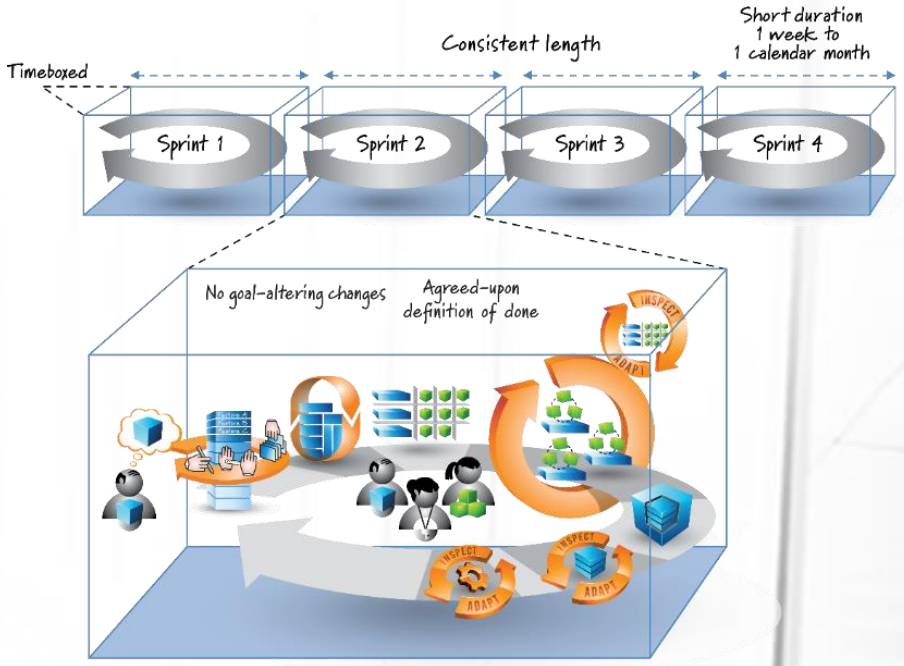
- Sprints reviews are checkpoints for stakeholders to provide feedback and be able to pivot if things aren't right.





Should the sprint duration be consistent?

- Absolutely! Each development project should pick a consistent duration for its sprints
- However, compelling reasons to alter the duration include:
  - Longer to shorter sprints to see if more frequent feedback would be better
  - Seasonal calendar situations (end of year)
  - Product release due in less time than the sprint
- Not acceptable reason – team needs more time to complete the work



Why do we want a consistent duration? Cadence!

- **Week long sprint** = 5 calendar **weekdays**, 2 weeks = 10 calendar weekdays, etc.
  - Any **holidays, training days**, etc. within a sprint just reduces the team's **capacity for that sprint**
- Promotes a **Cadence** (rhythm or "heartbeat") to the development work
  - "Get into the zone", "Get into flow", "Be on a roll", "Get into a groove"

Why do we want a consistent duration? Cadence!

- Levels out the intensity of the work
- Simplifies **Planning** and amount of work that can be completed, called **velocity**
- Velocity, the amount of total work that can completed in a sprint. Typically a full-time resource on a 2 week sprint has 8 story points (8 days out of 10 total days so 80% allocation).

The Sprint Goal

- Each Sprint has a clear **business purpose** and **value** which may be multifaceted (e.g., “Do this and that”)
- The Scrum Development Team should help refine and agree to the sprint goal during **sprint planning**
- **Sprint Goal:**
  - **Mutual Commitment** – Product Owner & Development Team
  - **Clarification** (not **Change**) is allowed but the difference between the two is such that a **change will have an impact** on meeting the sprint’s goal and work completion (done)

The Sprint Goal

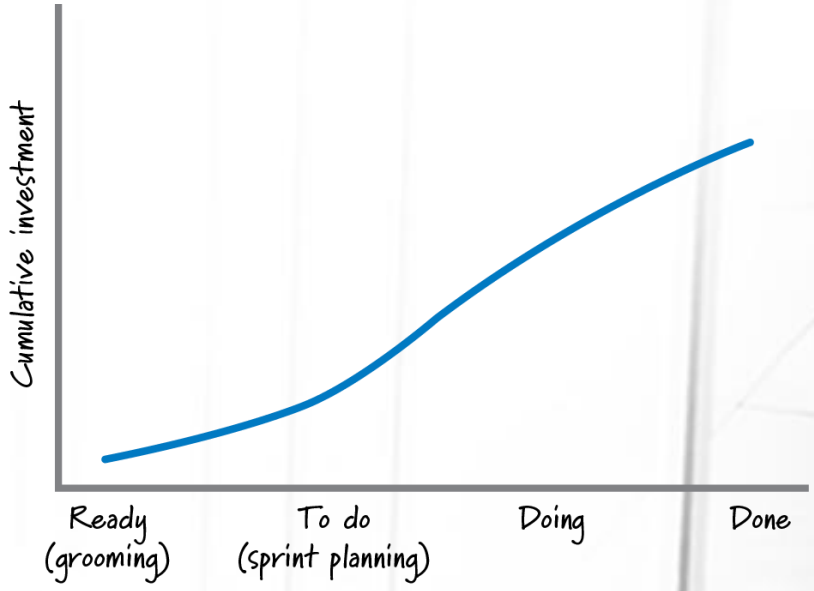
- This should be a discussing that occurs at planning with your team.
- Think about this:
- **What would be the goal of your first sprint?**

Clarification is allowed during the Sprint, change normally isn't

- Change (Add new functionality to implement):
  - PO: "When I said I wanted to search the police database for an offender, I didn't just mean by last name and first name. I also meant we should allow search based on pictures of suspect tattoos".
  - This is adding scope…Shouldn't be done as it impacts the planned work
- Clarification (Further defines how the existing functionality will work):
  - Dev Team: "When you said the matches for an offender search should be displayed in a list, did you want the list ordered a certain way?"
  - PO: "Yeah , sort them alphabetically by last name"

Why don’t we want changes during a sprint?

- Change has Consequences
  - Scrum principle embraces **Change** but in a balanced, economically sensible way
  - **Economic consequences** of a change **increase** as our **level of investment** in the **changed work increases** (Figure 4.6)
    - Initial Sprint Planning
    - Replan for the Sprint
    - Investment in work increases as backlog items progress from “to do” to “in progress” to “done”
  - Dev Team **Motivation & Trust** deteriorates

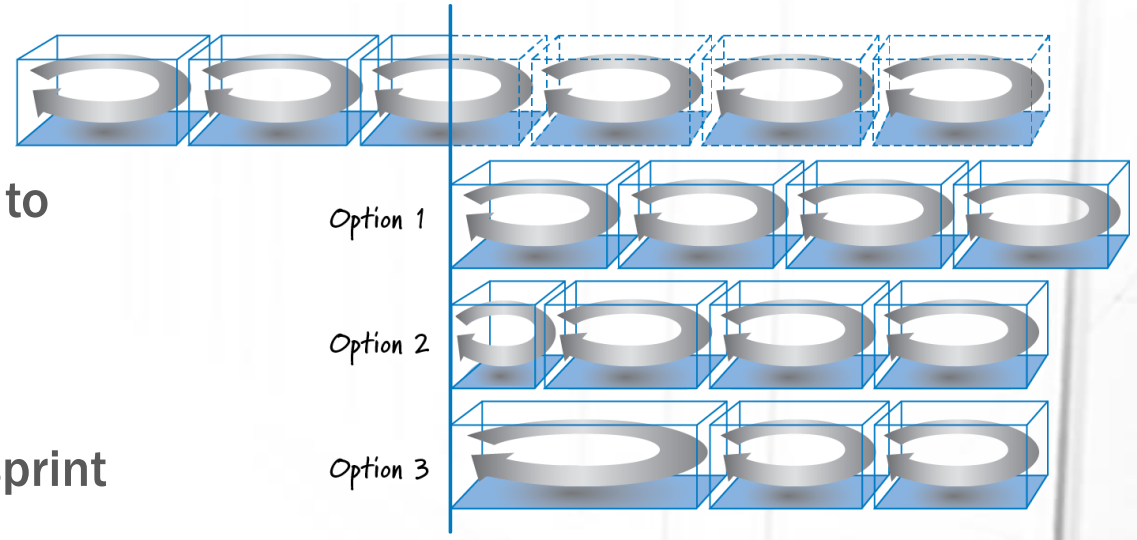


Why don't we want changes during a sprint?

- The **No Goal Altering Changes** characteristic is a **Rule** not a **Law** and being pragmatic "**trumps**" it
- Business conditions can necessitate changes to sprints
  - Correct business decision is to make the change if its consequences are significantly less than deferring the change and vice versa
- Immaterial consequences suggest to defer the change
- Emergencies can occur, but if your team is always dealing with emergencies then scrum may not be the right framework.

Terminating a Sprint

- **Abnormal Termination** of a Sprint occurs when it becomes completely invalid
- Sprint terminates immediately
- Scrum Team (PO, SM, Dev Team) meets to perform a Sprint Retrospective
- Team then plans a new sprint
- PO's reserve the right to terminate any sprint but doing it is a serious disruption
- Scrum Team decides length of the next sprint (Option 2 or 3 best for multi-team)



What to Expect when the sprint is completed? (Done)

- **Result** of a sprint is a **Potentially Shippable Product Increment**
- Actual deployment of the product increment is a business decision
- Sprint's result is a **state of confidence** that what got built is actually....done
- Conceptually, the definition of **Done** is a checklist of the types of work that the team must successfully complete for the **entire product increment**

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, Integration, Regression, Platform, Language
Zero Known Defects
Acceptance Tested
Live on Production Servers

Definition of Done

- **Definition of Done** (applies to the product increment) can evolve over time as organizational impediments or limitations may necessitate
  - **Earlier sprints** may have a definition of Done that is somewhat different than **later sprints** due to this
  - Leaving an activity out of a sprint (such as performance testing) could have a backwards ripple effect when that activity is actually performed
- Definition of **Done** versus **Acceptance Criteria**
  - Each product backlog item in a sprint should have a set of **conditions of satisfaction (acceptance criteria)** for the Product Owner
  - Acceptance criteria are item specific and in addition to definition of Done
  - **Completed** or **Accepted** (not done) are terms used when Product Backlog items pass their acceptance criteria

Sprint Backlog

- The Product Owner maintains a groomed backlog of the items that need to be worked on.
- During the planning meeting, the product owner discusses the highly prioritized items and the team decides what they can work on.
  - Maintaining the correct priority is important. Choosing to do the right prioritized work can make a significant impact!

Sprint Backlog

- The highly prioritized stories that are estimated in the planning meeting (or possibly before) and have been decided to be worked on….
- ….then officially move from the product backlog to the sprint backlog.

Product Backlog -> Sprint Backlog (Its now committed work)

- The sprint backlog contains all the stories that the team will work on during that given sprint.

Execution during a Sprint

- Once a sprint is underway, there should be minimal disruptions to the sprint.
- The team should be focused on the sprint goals and completing their tasks.
- The team has an agreed upon planning from the organization!

Execution during a Sprint

- During the sprint, the team is essentially self-organized, and they will so whatever is necessary to achieve the sprint goals stay product focused.
- The Scrum master assists to remove impediment, but they should not act as a manager.
- The ownership should be on the team to complete their stories.

Execution during a Sprint

- Think about this:
  - **Why is it important for the scrum master to not be their manager?**
  - **What are some of the things that could happen if the scrum master becomes a people manager?**