ECE 420

Lecture 7

October 14 2019

Software Development Practices

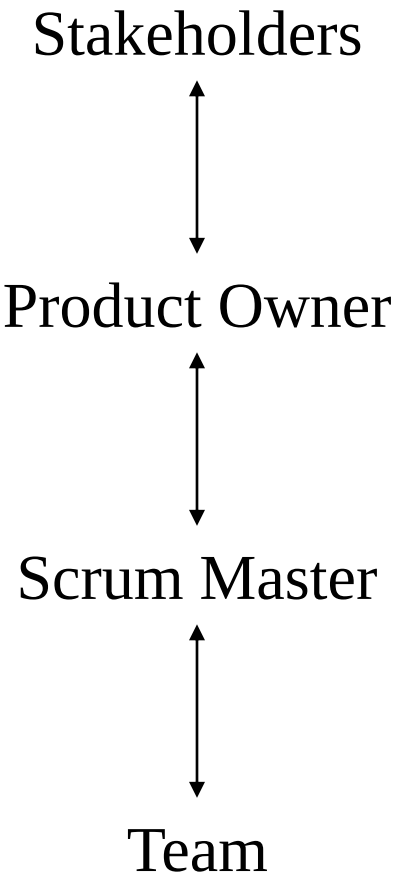
- Structured methodology for the planning, execution, delivery, and maintenance of software products
- Absolutely essential for large companies / teams
- Many, many different models
  - Agile, waterfall, spiral, among others
  - Select the one with the best fit for the project / company / application space
  - Teams might implement a model with modifications, or use a hybrid of various models

Agile / Scrum

- Transparency and visibility of development and goals
  - Focus on highest priority / most important tasks
- Self-organizing team process
  - Scrum is a lightweight framework not an explicit set of detailed rules
- Work performed in short iterations (sprints)
- Incremental improvement
  - Potentially shippable product at end of any sprint
- Frequent reassessment facilitates adapting to change
  - Modifications or additions to requirements by customers / management
  - Changes necessitated by discoveries during development
  - Identifying issues and improving internal process

Roles

Roles



Stakeholders

- People interested/invested in the outcome of the team's efforts
- Internal stakeholders
  - Management
  - Product Development
  - Sales
  - Other developers
- External stakeholders
  - Customers
- Objective of the team is to make their Stakeholders happy (within reason)

Product Owner

- Work with stakeholders to establish desired deliverables, needs, and priorities
  - “Product” may be an actual product, a feature, a tool, etc.
- Work with development team to define tasks (stories) to achieve deliverables
- Create, organize, update and prioritize stories, directing team's overall objectives
- Monitor progress
- Respond to questions and issues
- Accept or reject completed work
- Not necessarily the team’s formal Manager

Scrum Master

- "Coach" that helps make sure everyone is following the Scrum process and fulfilling their roles
  - Point of reference for best practices in Scrum
- Interface between Product Owner and Team
  - Monitor team productivity (velocity), assist in task selection
- Helps team update and improve their internal process
- Assist in addressing impediments that the team is not able to resolve themselves
- Might possibly serve as Scrum Master to several teams

Development Team

- Usually 5-10 people (two pizza rule), usually doesn't change frequently
- Cross functional team, tasked with end-to-end development responsibilities
  - Design, implementation, and testing
- Sometimes PO, SM are team members, though scrum encourages these to be separate
- Works with PO to develop tasks, estimate effort
- Executes the accepted work for each sprint
- Self-organizing in terms of internal process
  - Scrum tells you what needs to be done but not how to do it

The Sprint

The Sprint

- A fixed duration (*timeboxed*) unit of work
  - Usually 1-2 weeks, max 1 month
- Work accepted at the beginning of the sprint is expected to be completed during the sprint
  - Enforces the prioritization of work
  - Displays project progress, frequent checkpoints
- Serves as a unit for more predictable planning
  - Easier to plan for
  - Fast feedback on project objectives
- Goal of completing all tasks during the sprint is not to add stress but to motivate the team

Sprint Meetings

- 5 main types of meetings
  - Sprint Planning
  - Standups
  - Sprint Review
  - Retrospective
  - Storytime
- Goal is to make meetings meaningful use of time and minimize impact on development efforts

Sprint Planning Meeting

- Kick off meeting for the sprint
- 1 hour, maybe less
- Development team, SM, PO attend
- Review the prioritized backlog of stories
- Select stories that the *team* is going to *commit* to for the current sprint
  - Sprint commitment should not change once sprint starts
- Ideally every sprint will result in an incremental improvement in the team's product (though not necessarily shippable)

Standup

- Daily Status Meeting
- 15 minutes or less
- Dev team, SM, PO attends
- *Brief* update from previous day's work
  - Progress on stories
  - Issues/blockers encountered
  - Focus for today
- Deeper technical discussions reserved for post-meeting
- Usually useful to have towards beginning of work day
- Keeps everybody aware of team progress on sprint goals, if particular stories are *at risk*

Sprint Review Meeting

- End of sprint meeting
- About 1 hr, depending on size of team and format of review
- Open to all (stakeholders in particular)
- Review of work completed in current sprint
- PO formally approves or rejects completed work stories
- Development team provides a demo / brief code overview / other gathered statistics
  - Not a detailed code walkthrough
  - Enough to convey work was completed successfully

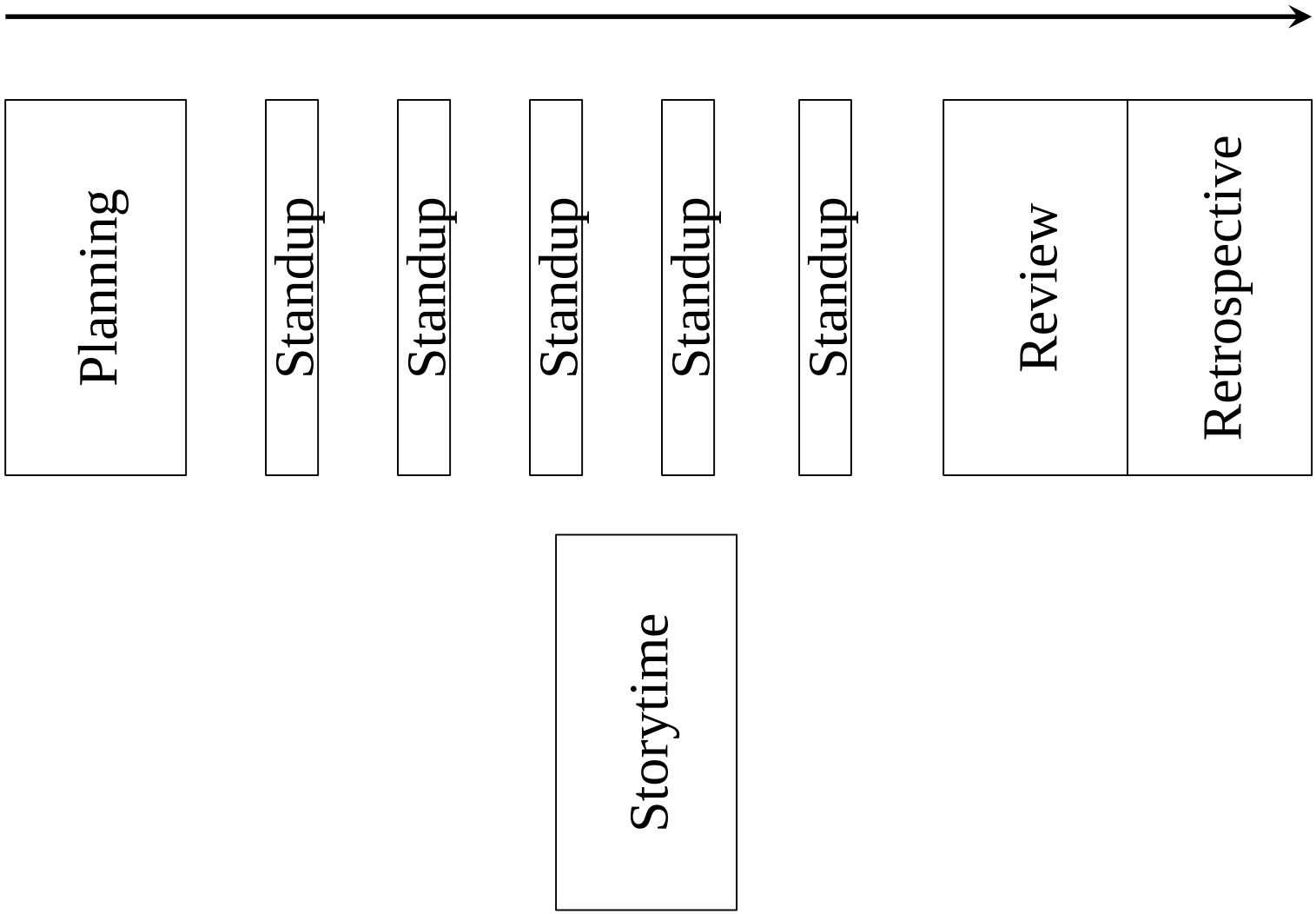
Sprint Retrospective Meeting

- “Between sprints” meeting, but can happen at any time
- About 1 hr
- Dev team, SM only
- Objective is to have a discussion about previous sprint
  - What went well
  - What didn’t go well
  - How to improve what didn’t go well
  - Other things that could be done differently / more effectively
- Form an Action Plan to address raised issues
- Revisit at next Retrospective

Storytime Meeting

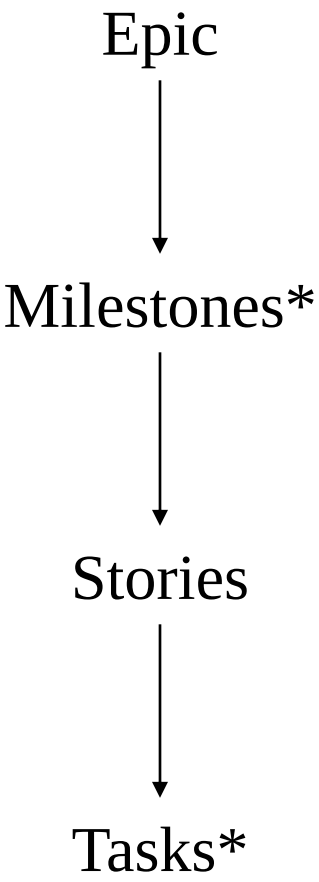
- Sometime during the sprint
- Development Team, PO
- Objective is to write, revise, and estimate stories
- Make sure that stories are well defined and ready for prioritization
- Stories can be written outside of this meeting and reviewed here
- Could be an optional meeting, but makes the Sprint Planning meeting much easier

Sprint Overview



Stories

Story Hierarchy



(User) Stories

- An activity/goal as a step toward achieving a larger project aim.
- Includes a high level description understandable to technical and non-technical people.
- Canonical form story
  - As a <role> I would like <goal> so that <benefit>
- Contains Acceptance Criteria (or Definition of Done)
  - Measurable and independent way or assessing story tasks are completed
- Establish any prerequisite stories or other preconditions (story Readiness)

Stories

- Does not contain implementation details but does contain enough information about task that effort can be estimated
- Sized appropriately small
  - Couple days max generally
  - Large stories run risk of not being completed during sprint
- Stories may be annotated with sub-tasks
  - More technical, usually added during sprint planning
- How to estimate story effort?

Story Effort

- Common method is to assign points
  - Power of 2 (1, 2, 4, 8, ...) or Fibonacci numbers (1, 2, 3, 5, 8, 13, ...)
- These do not correspond to time commitment but establishes relative complexity
- Idea is leverage past experience to judge story relative to past tasks and rank accordingly
  - Similar to something we did before?
  - Easier?
  - Harder?

Story Effort

- Planning poker
  - All team members make their assessments and reveal jointly
  - Try to judge more 'instinctively' about effort without over-thinking / over-analyzing
  - Don't get biased by others' opinions
- If story is estimated very high point value (e.g. 21+ points), then the story is too large
  - Cannot fit within the sprint or reasonable about of time to manage
  - Large size indicates some uncertainty about accurate scope
  - Break up into multiple smaller stories

Project Planning

Product Backlog

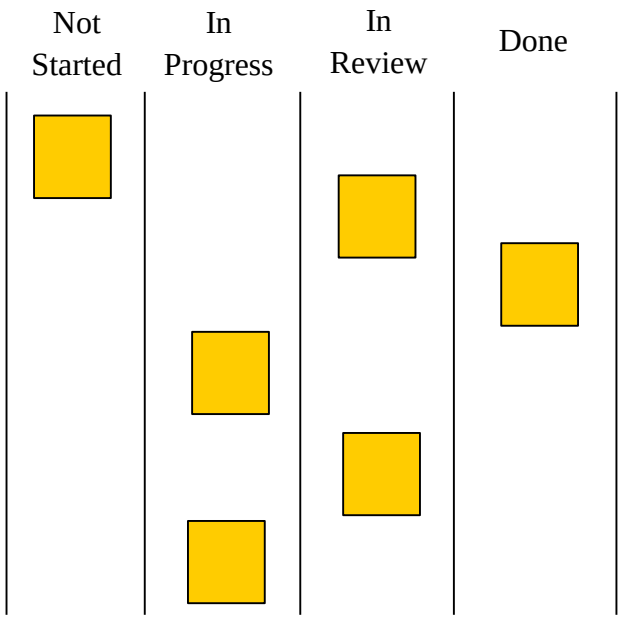
- All the 'todo' items on a project
  - Contains stories relating to various aspects/features of the project
- Stories grouped into Epics relating to over-arching feature delivery
- Prioritized by the PO with most important stories on the top
  - Reprioritized as new stories are added and at the end of each sprint
- Stories added by PO or Developers

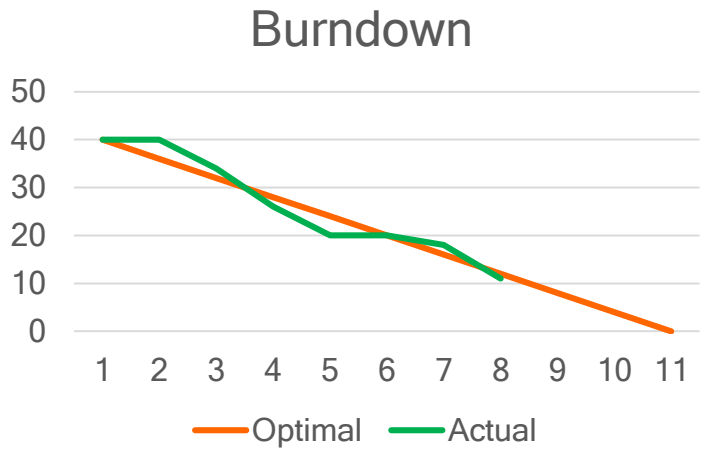
Project Planning

- An epic representing a particular deliverable can be broken down into its constituent stories
- Each of those stories can be assessed and estimated
- Over time, the team will establish a *velocity*, or number of points per sprint that can be reliably undertaken
  - Also point estimation tends to become more consistent over time as a larger pool of past work becomes available to compare to
- Total estimated time to complete a project
  - Total Points / Velocity * Days per Sprint

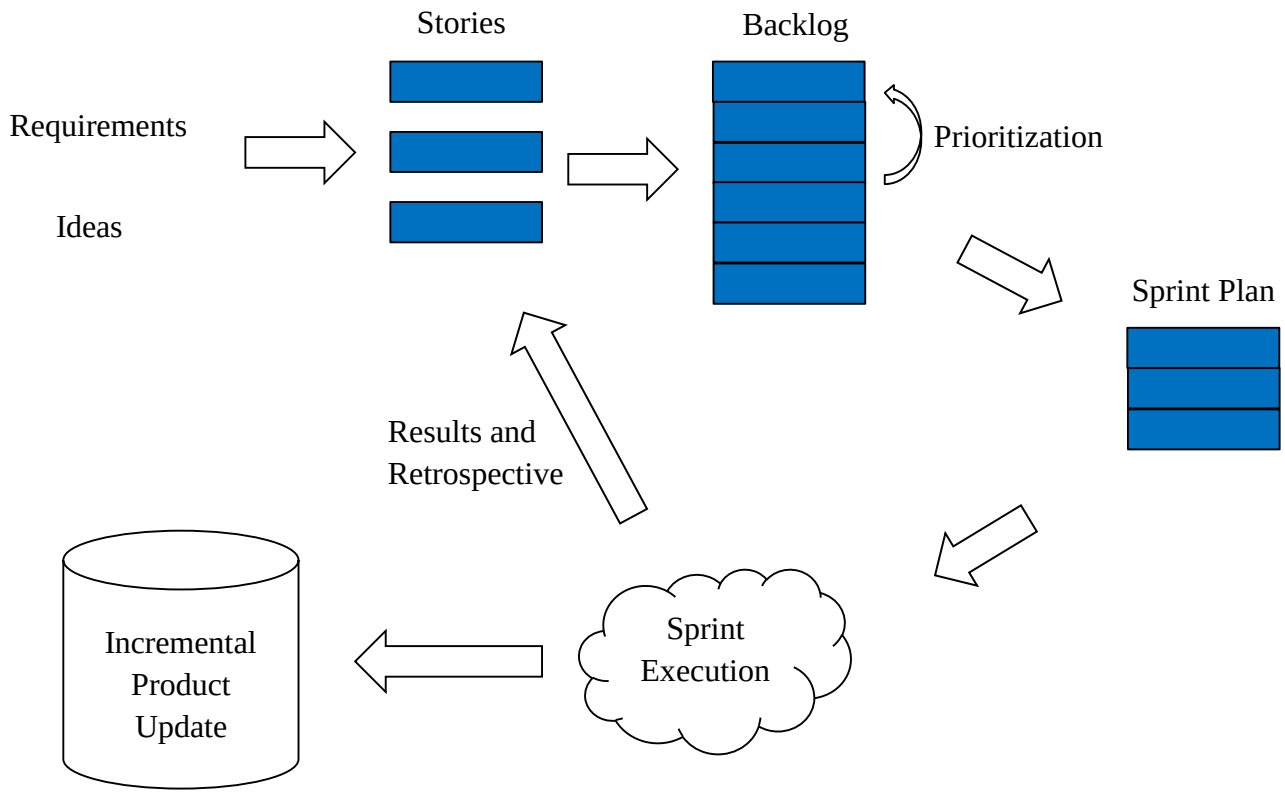
Project Monitoring

- Scrum Board
  - Physical or Electronic Tracker for Story/Subtask State
  - Team members update in real-time
  - Assess sprint state, active/done/TODO items
  - Used in standup
- Burndown chart
  - Plot of story points not completed vs. time
  - Graphical representation of progress, if sprint is at risk





Scrum Pictorial Overview



Summary

- Scrum is one of many methodologies for managing the software lifecycle
- Some particular benefits of scrum
  - Adaptation to change (internal or external)
    - Constant rebalance of priorities with focus on shippable results
  - Fosters teamwork and team empowerment
  - Encourages distribution of knowledge
  - Built-in self-improvement mechanism through retrospection
- No methodology is a perfect fit – adapt as needed!

Using Scrum?

- How could Scrum help on the Final Project?
- High level planning
  - Breaking down deliverables into stories, estimating difficulty, prioritizing work
- Project execution
  - Coordinating work, clarity on objectives
- Periodic Reassessment
  - Tracking progress week-to-week, adjusting goals and process
- Above all, communicate with your team members and do what works best for your team!

This week

- Lab 7: Video Tracker Quiz/Demo
- Assigned Project Labs