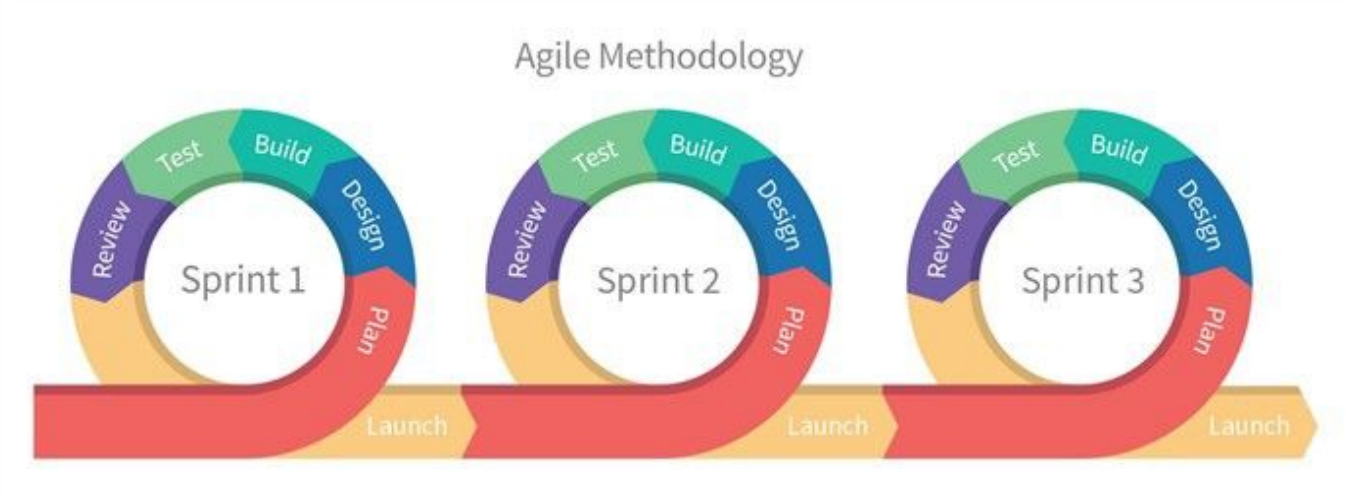
Lecture Notes
October 9, 2018

Team assignment Revised Project Proposal is due tonight, 11:59pm.

Make sure your IA mentor knows what you're doing, but don't wait for grading, start developing asap!

Next team assignment First Iteration due Tuesday, October 30, 11:59pm

Follow-up team assignment First Iteration Demo due Thursday, November 8, 11:59pm

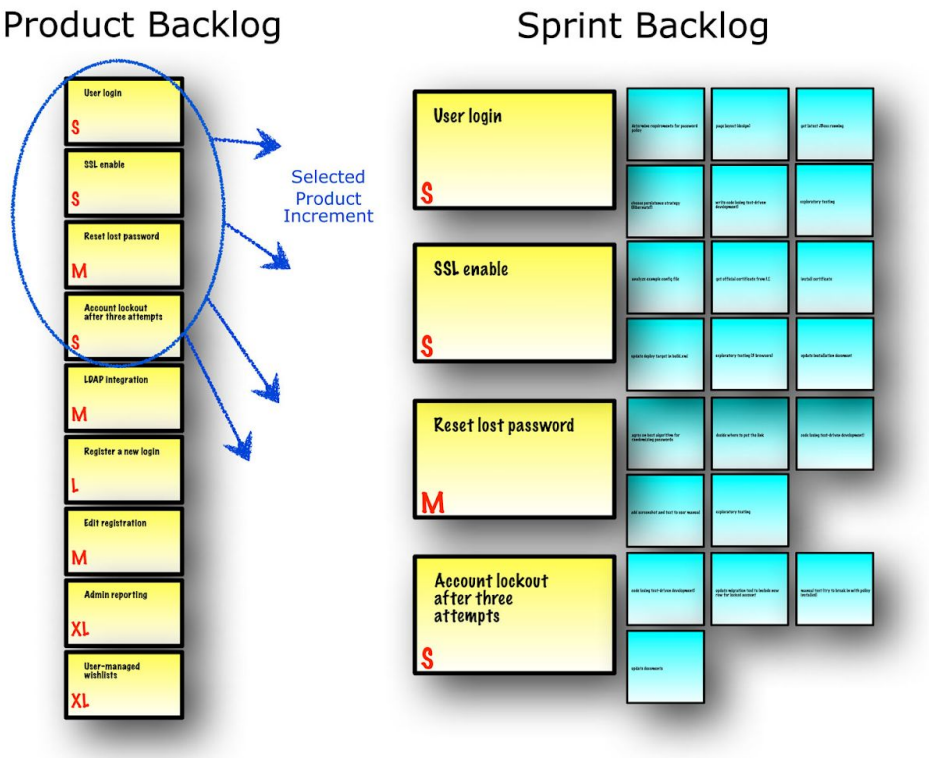


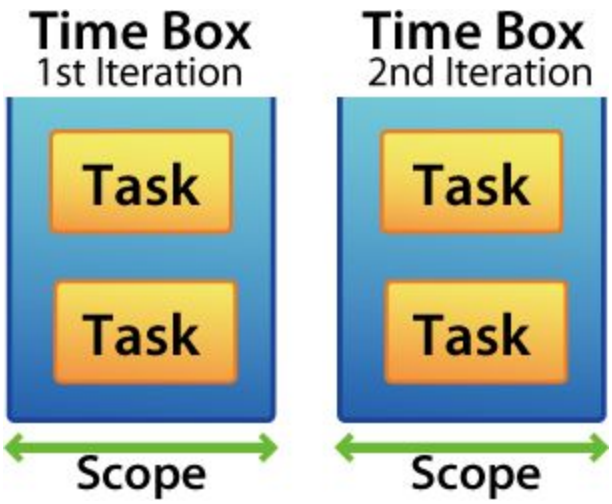
An agile project builds the software product in a series of iterations (sprints).

Ideally, a new version of the product is delivered to customers after every iteration (or even after every successful build for devops and other in-house projects), but typically it takes several iterations to build the first release

Planning for first release: Start with minimum viable product (MVP) set of user stories, and fit roughly into a series of iterations

For a new project (that is not very similar to some previous project), initial schedule may be pure guess





Since iterations are fixed-length (typically two weeks), there are a maximum number of hours/days available =

# of developers (or pairs)
x 10 working days
x 8 hours/days

Or # developers x 9 x 7 - plans time for half-day pre-sprint planning at beginning, half-day post-sprint review and retrospective at end, plus daily standups and other meetings

The developers need to *estimate* how long each user story will take to complete, to see if all those grouped for the first iteration will really fit

Estimates may come from previous experience, e.g., formal or informal metrics from past projects

Agile estimation approach is "planning poker" - playing cards marked ½, 1, 2, 3, 5, 8, 13, 20, 40, ∞, ? days

Very small stories grouped so minimum size is ½ day

For each story, each developer independently plays a card - then team discusses and comes to a consensus

Many organizations use relative "story-points" instead of absolute time - 1 point for smallest/simplest stories, 3 points for a story that seems about 3 x as long/difficult, etc.

Then convert story-points to days/hours afterwards (how long did it really take) to guide future planning

Even when using time-based estimation, still recalibrate at end of every iteration, e.g., if estimated ½ day really took 2 days, estimated 1 day really took 4 days, etc., need to multiply estimates by 4

If wide divergence among developers, something is wrong

Estimates longer than 8 days (essentially an entire iteration) are also suspect:

- ➢"I have no clue how long this will take"
- ➢Probably need more information from customer and/or break down into smaller user stories

Spike solutions for longer/riskier stories - invest ½ day or full day on quick and dirty *throwaway* version separate from main line of development, to better estimate how long real version will take

After time estimates assigned to every user story for first iteration, cut down scope so time estimates total to no more than 70% (or 80%, etc.) of time available

Why not 100%? *Project velocity* accounts for lost time

If team runs out of work to do, add more stories from backlog

If team cannot finish everything, move some scheduled stories to backlog

How many estimated time units or story-points did team really finish and demo/deliver? This is the project velocity for that iteration

Why do we need project velocity?

Computers have lives (hardware and software maintenance and downtime)

Developers have lives (personal days, sick days, unproductive days)

Projects have lives (training, meetings, customer interruptions)

Organizations have lives (CTO unexpectedly schedules full-day all-hands meeting)

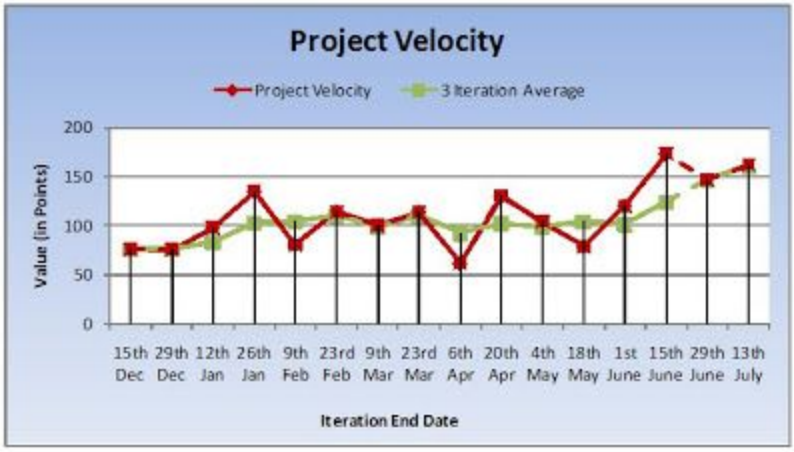
Example (true story): company tells new hire that the work week is 10am-6pm Mon-Fri, the supervisor gives new hire 32 estimated hours of new work each week, plus there's a one-hour department-wide meeting once per week and two half-hour team meetings per week

Is there a problem?

At end of each iteration or beginning of next iteration, developers re-estimate for remaining MVP features (and customer may reconsider what must be included in MVP - emphasis should be on *viable* not *minimal*)

Use project velocity from Nth iteration to plan schedule for N+1th iteration

OR use average project velocity of several recent iterations



Planning for later releases: Customer re-prioritizes remaining / newly added user stories, team re-estimates (with greater knowledge) for next set of iterations

How do testing and debugging fit into iterations?

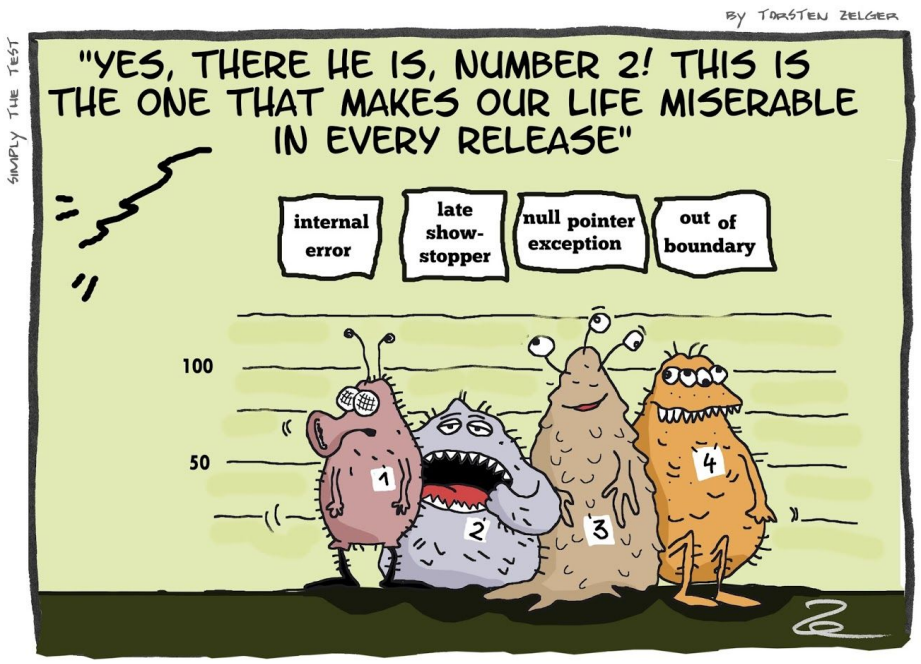
Unit testing (and corresponding debugging) are part of completing a user story. Formal code review often also required before considered "done". Need to include this time in estimate

Bugs may be reported by separate testing team (with separate iteration cycle), typically at planned times, or by users at any time. Debugging these bugs defined as special kind of user story (e.g., pink cards) - also prioritized and estimated

Fit show-stopper bug stories into current iteration by pushing some previously planned stories to backlog, to be considered for next iteration

Include remaining highest priority bug stories in later iteration just like new features

There is often a "hardening" iteration as last iteration before release, with only testing and bug fixes. Many organizations also devote one day per iteration, or one day per week, to debugging



Lining up for the aftermath session

The above is about "engineering sprints". Engineering sprints are sometimes preceded by or interleaved with Design sprints

Concept developed by Google Ventures (GV) for their startup portfolio

> Many GV companies are **not** in the software industry, apply to any industry
>
> Intended to answer business questions without full development and launch cycle

0th day (ahead of time) - assemble team, get space, buy supplies, etc.

Core team members need clear schedules for week

Facilitator runs meeting throughout, *Decider* makes ultimate decision in any debates

Monday morning

"Start at the end" - what is the long-term goal? What are the risks?

"Map the challenge" - workflow sketch of customers/key players interacting with product

Monday afternoon

"Ask the experts" - bring in a few domain and business experts for short interviews

"How might we" - approach to asking questions and taking notes during interviews, reframe problems as opportunities

After experts leave, sort notes into categories

Pick target to focus on during rest of design sprint

Tuesday

Review existing ideas (e.g., from other products) to remix and improve

Team members separately sketch ideas, either competing alternatives or dividing up the problem

"Crazy 8s" - set of 8 quick sketches in 8 minutes

Start recruiting customers for tests to occur on Friday

Wednesday

Go through sketches, critique each, choose best

Weave into storyboard and fill in gaps (or pick two or three for "rumble")

Thursday

Implement “fake it” prototype (facade, possibly just marketing materials)

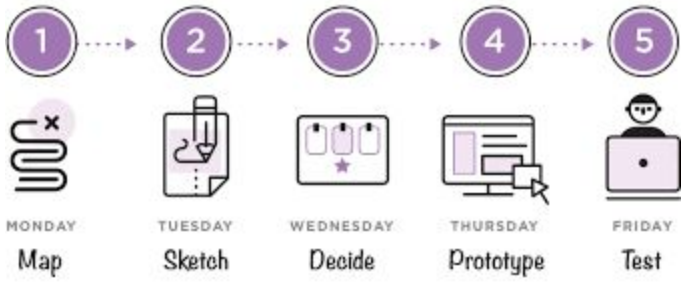
Trial run - interview script

Friday

1-on-1 interviews with 5 “customers”, show prototype, open-ended questions

Rest of team watches video feed and takes notes

Review “customer” patterns - positive, negative, neutral - and decide how to follow up



In-class exercise if time permits:

Sit with your team and conduct a mini design sprint for your project

5 minutes

What are the risks?
Map the challenge - user workflow

8 minutes

Each team member fold paper into eight sections
Sketch an idea in each rectangle

5 minutes

Discuss sketches, decide which “best” or “useful”

Read for Thursday:
textbook, chapter 4 Putting Testing First

Next pair assignment, Bug Hunt due next Tuesday, October 16, 10:10am.

Next team assignment First Iteration due Tuesday, October 30, 11:59pm. Follow-up team assignment First Iteration Demo due Thursday, November 8, 11:59pm.