Lecture Notes
September 28, 2017

Please sit together with your teams

3rd individual/pair assignment (finding bugs) due tonight

4th individual/pair assignment (development technologies) due Tuesday October 3

3rd team assignment (revised proposal) due Thursday October 5

Design Sprints

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

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

[Design Sprint Explained in 1 Minute]

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 a 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,
> possibly dividing up the problem rather than competing

“Crazy 8s” - set of 8 quick sketches, then three-panel storyboard

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)

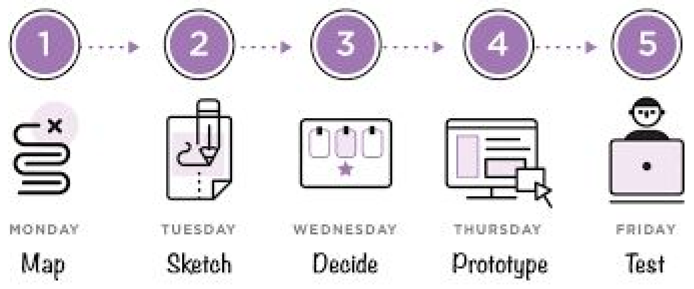
Trial run - interview script

Friday

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

Rest of team watches video feed and takes notes

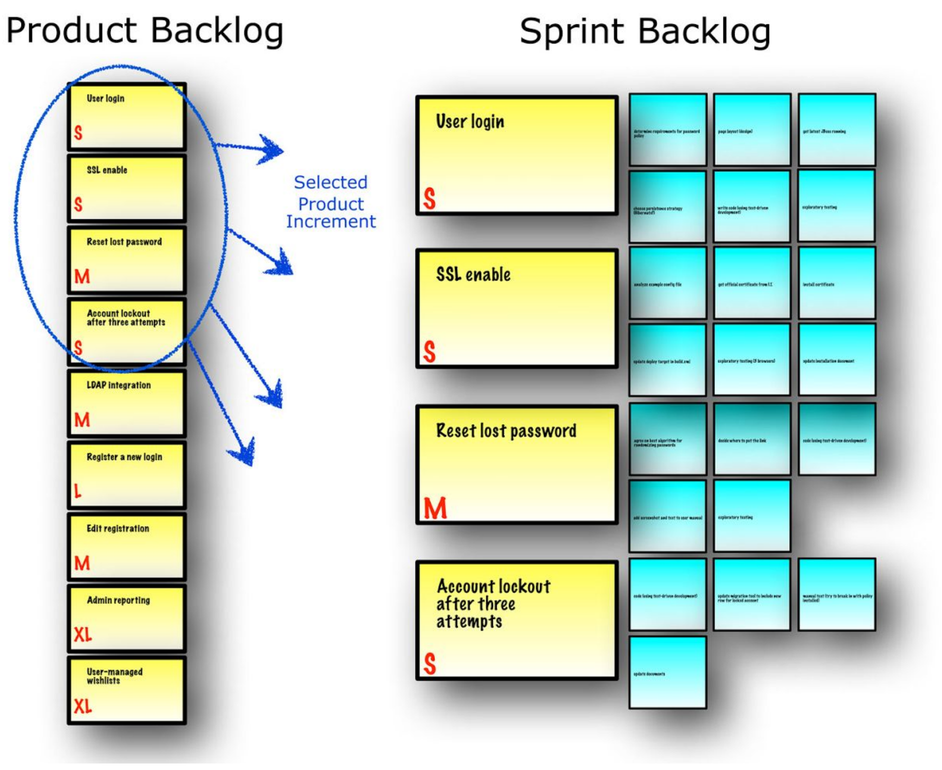
Review “customer” patterns - positive, negative, neutral - and decide how to follow up



mini design sprints for project during class time

Project Planning

We have initial requirements from the customer in the form of user stories and/or use cases, this is what some agile processes call the “backlog”, what next?



Plan the project one iteration (“sprint”) at a time

Possibly 1-week “design sprint” at or before beginning of project, possibly several devoted design sprints during long-term projects

Iteration / engineering sprint might be 1 week, 2 weeks, 1 month - most commonly 2 weeks

Per-sprint planning time at beginning ½ to 1 day (partially with customer or product owner)

Per-sprint review time at end ½ day or less, includes demo (with customer, other stakeholders)

Separate per-sprint retrospective ~1 hour (without customer or product owner)

First release of new product might take several sprints, e.g., 3 months (fiscal quarter), where delivery may require its own release sprint rather than fitting into per-sprint review

During iteration planning time, consider set of highest priority user stories not yet completed

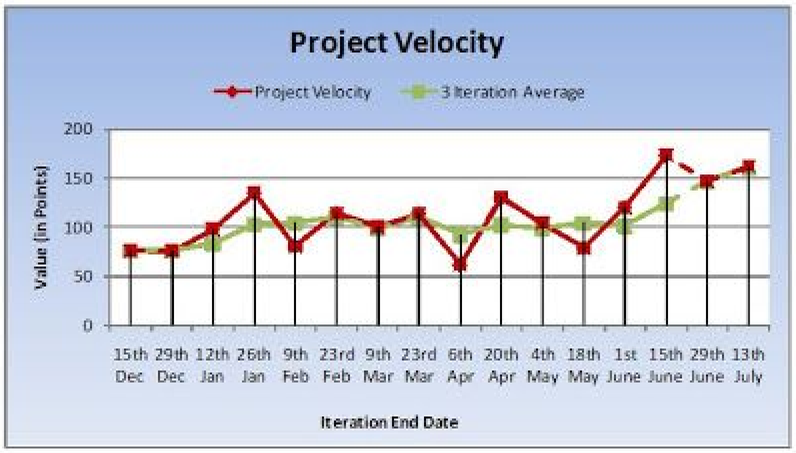
> Time or “story point” estimate for each - story point might be based on complexity not necessarily time
>
> How many points will fit in the iteration, considering project velocity?

Project velocity accounts for lost time - although “recommended” starting at 70% and recalculating at beginning of each iteration, many companies use 80% throughout

> If your team “should” be able to do, say, 100 points during an iteration, at beginning of first iteration schedule only 70 or 80
>
> 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 points did team really finish and demo/deliver? This is the project velocity

At end of each iteration (or beginning of next iteration), customer or product owner reprioritizes and developers reestimate times/points for remaining high priority features

> Use project velocity from Nth iteration to plan schedule for N+1th iteration
>
> OR use average project velocity of recent iterations



Priorities come from the customer

Minimum viable product

High, medium, low - may change over time

Typically there will be more higher priority features than will fit in iteration, remember time box vs. scope box

Possibly micro-prioritize within priority category (buckets 10, 20, 30, 40, 50)

Time/point estimates come from developers (or developers' supervisors)

How many time units to complete a given user story (typical minimum unit = ½ day)

Completion of a user story includes wireframes, design, coding, testing, review

Often use "points" or other relative units instead of explicit time, X seems likely to take ½ as long as Y or 3 times as long as Z

Schedule time/points for preparing and doing demo/delivery too!

How do developers determine estimate? (Do not ask the customer!)

Previous experience (formal or informal metrics from past projects)

"Planning poker" - playing cards marked ½, 1, 2, 3, 5, 8, 13, 20, 40, ∞, ? days

	PLANNING POKER There's always someone who shows their cards too early...

Estimates larger than 8 days (2 weeks with time for meetings, demos, disasters) are suspect: "I have no clue how long this will take"

Spike solutions for longer/riskier stories - invest ½ day or full day on quick and dirty version separate from main line of development (plan to throw away)

Why is project velocity needed?

Predicting how long it will take developers to do work can never be 100% accurate

Computers have lives (hardware and software maintenance and downtime)

Developers have lives (personal days, bad days)

Projects have lives (training, meetings, customer interruptions)

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

Example: 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 (wannabe standups where they sit down), is there a problem?

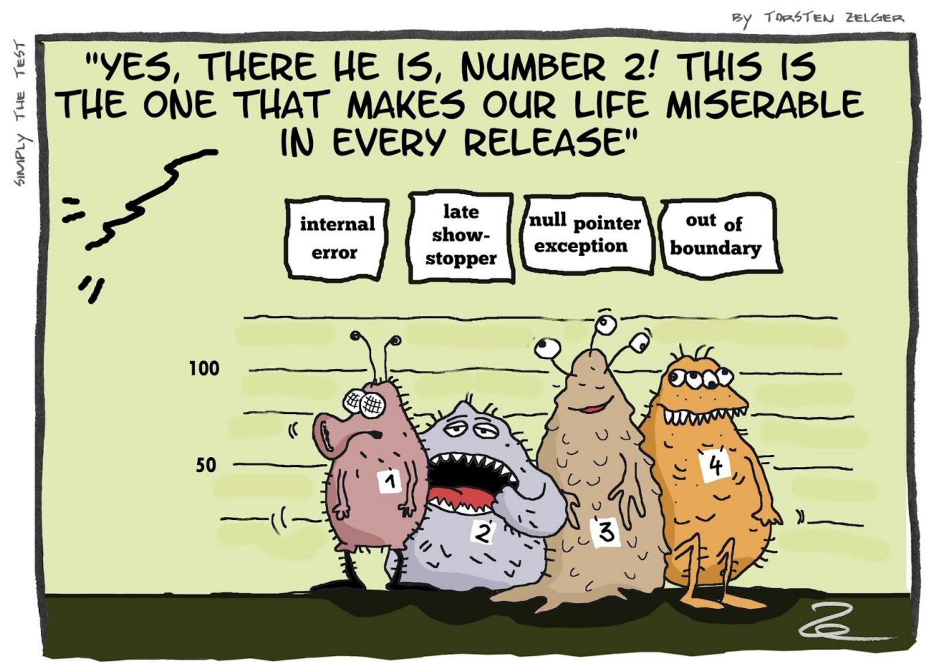
How do testing and debugging fit into iteration?

Unit testing (and corresponding debugging/fixing) part of completing user story, formal code review often also required before considered "done", include in estimate

Bugs may be reported by separate testing team (with separate iteration cycle), usually at planned times, or by users at any time

Debugging/fixing can be 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 next iteration (overflow)



Lining up for the aftermath session

Include remaining highest priority bug stories in next iteration (instead of new features)

In some organizations, every 3rd or 4th iteration devoted to debugging, or one day of every iteration is set aside for debugging and not included in available time for iteration