

Lecture 4: Intro To Process

Michael Hilton and Rohan Padhye

Learning Goals

- Recognize the importance of process
- Understand the difficulty of measuring progress
- Identify why software development has project characteristics
- Use milestones for planning and progress measurement
- Understand backlogs and user stories
- Know your team!

PROCESS

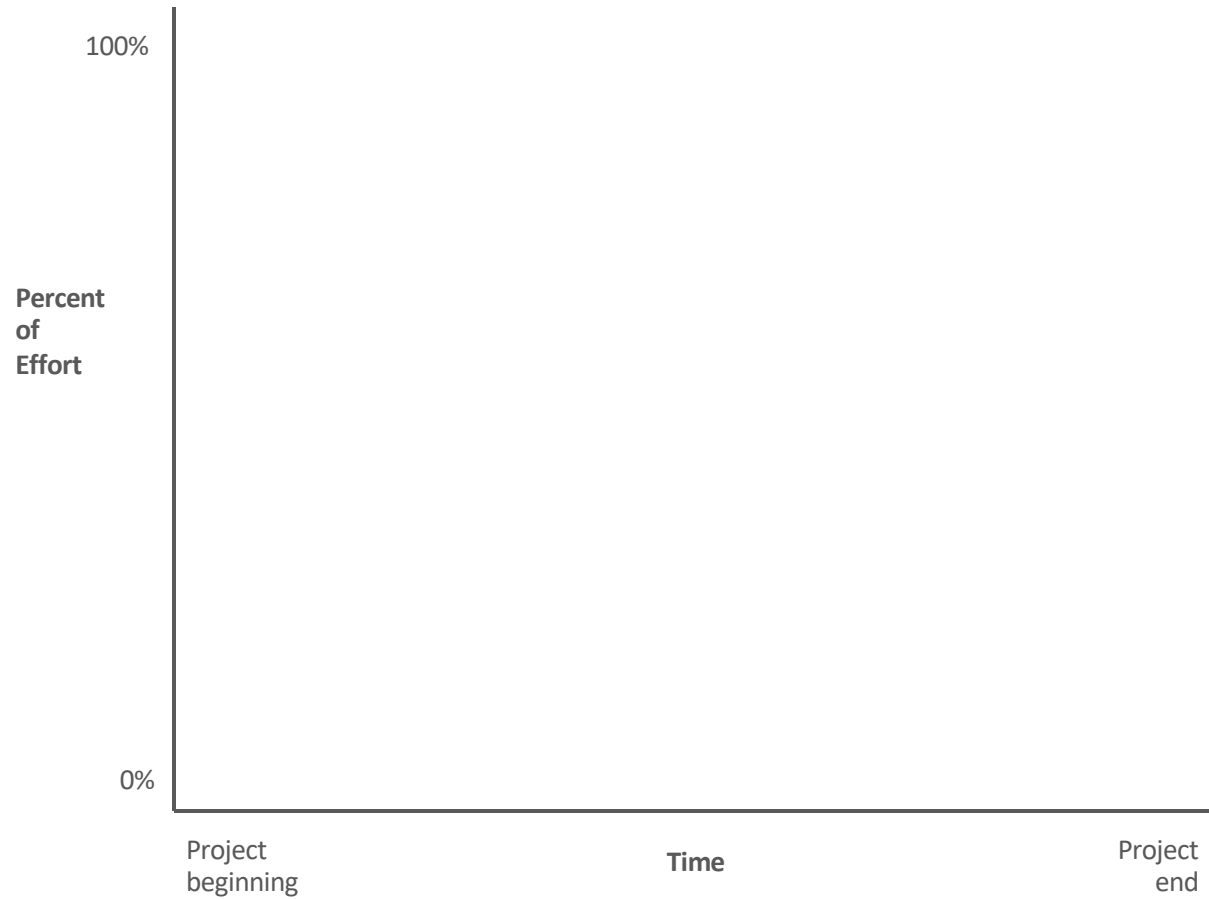
Software Process

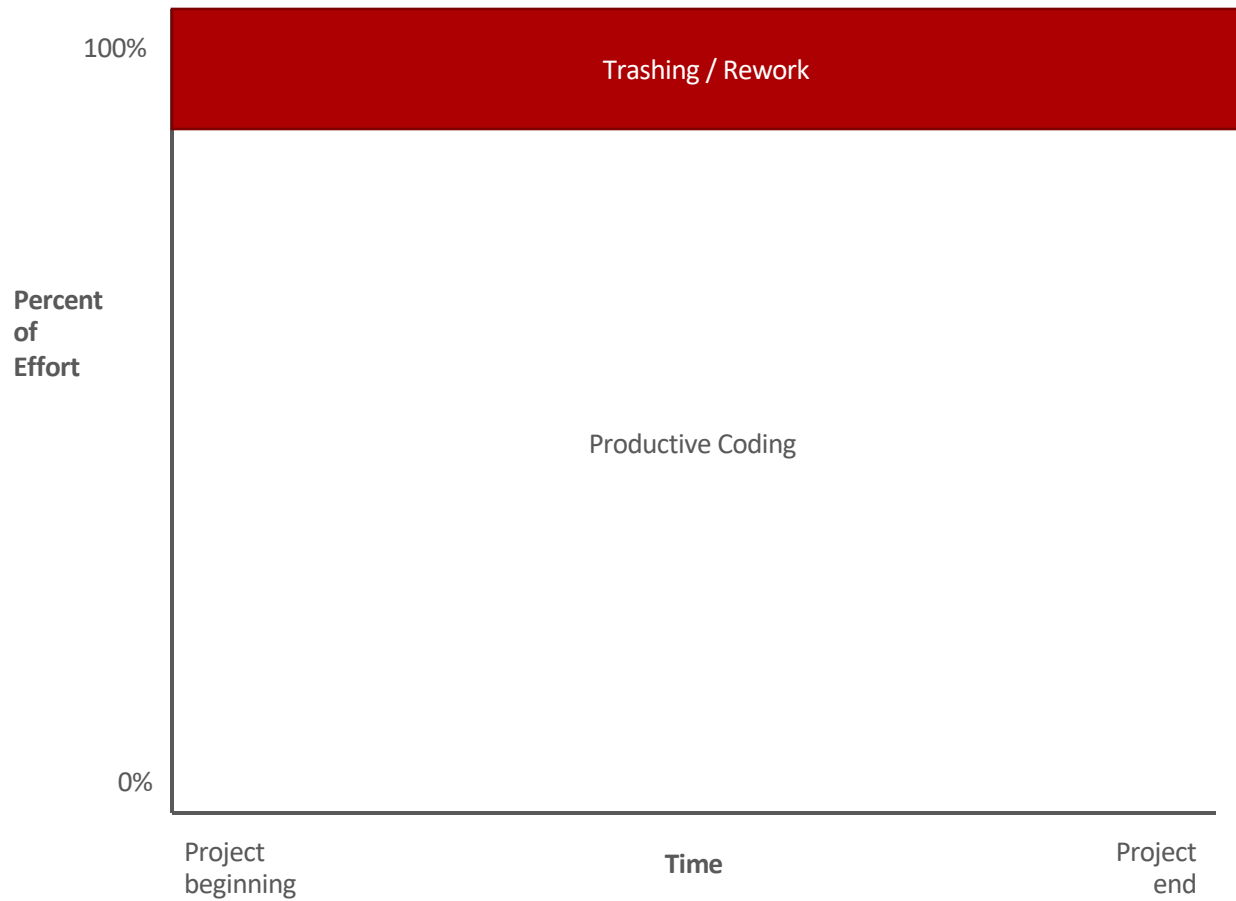
“The set of activities and associated results that produce a software product”

Sommerville, SE, ed. 8

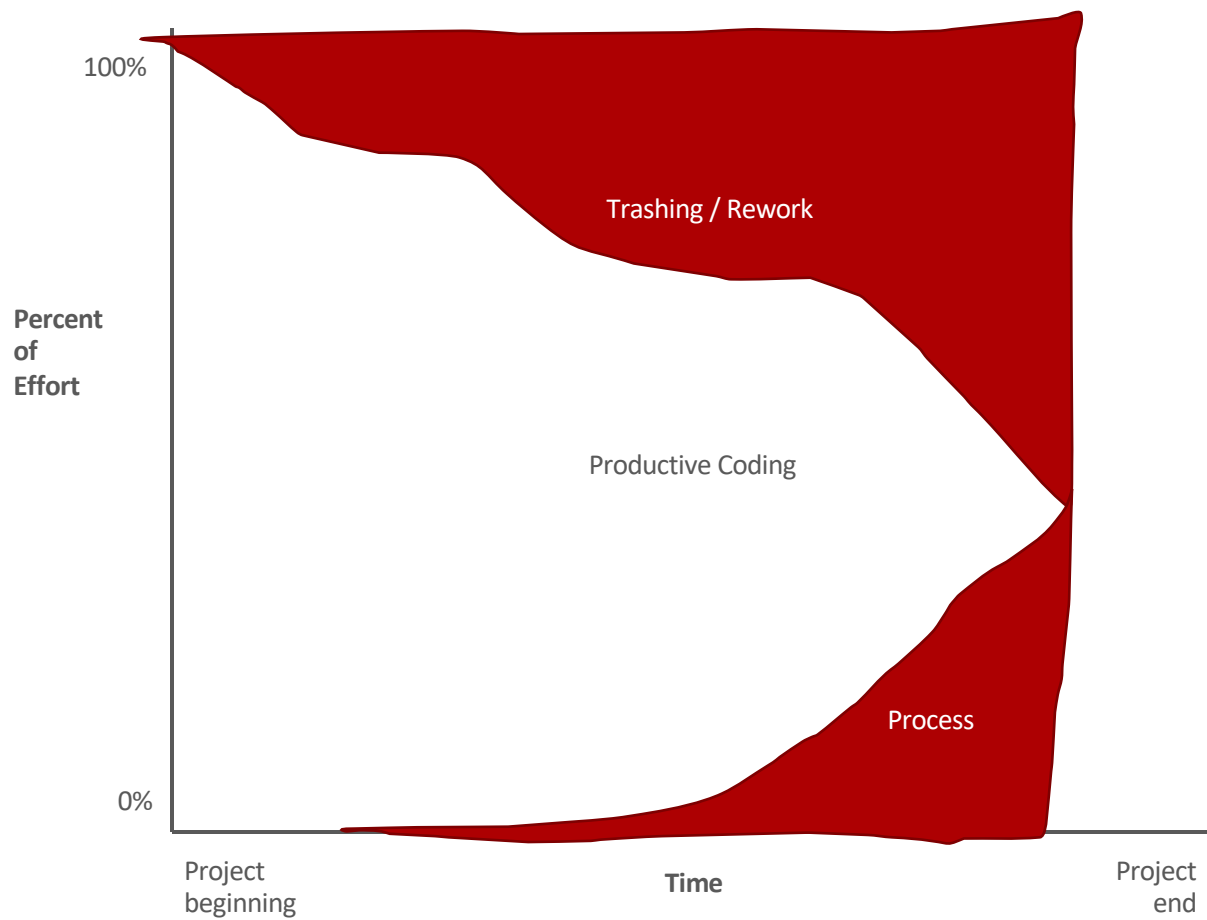
How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1







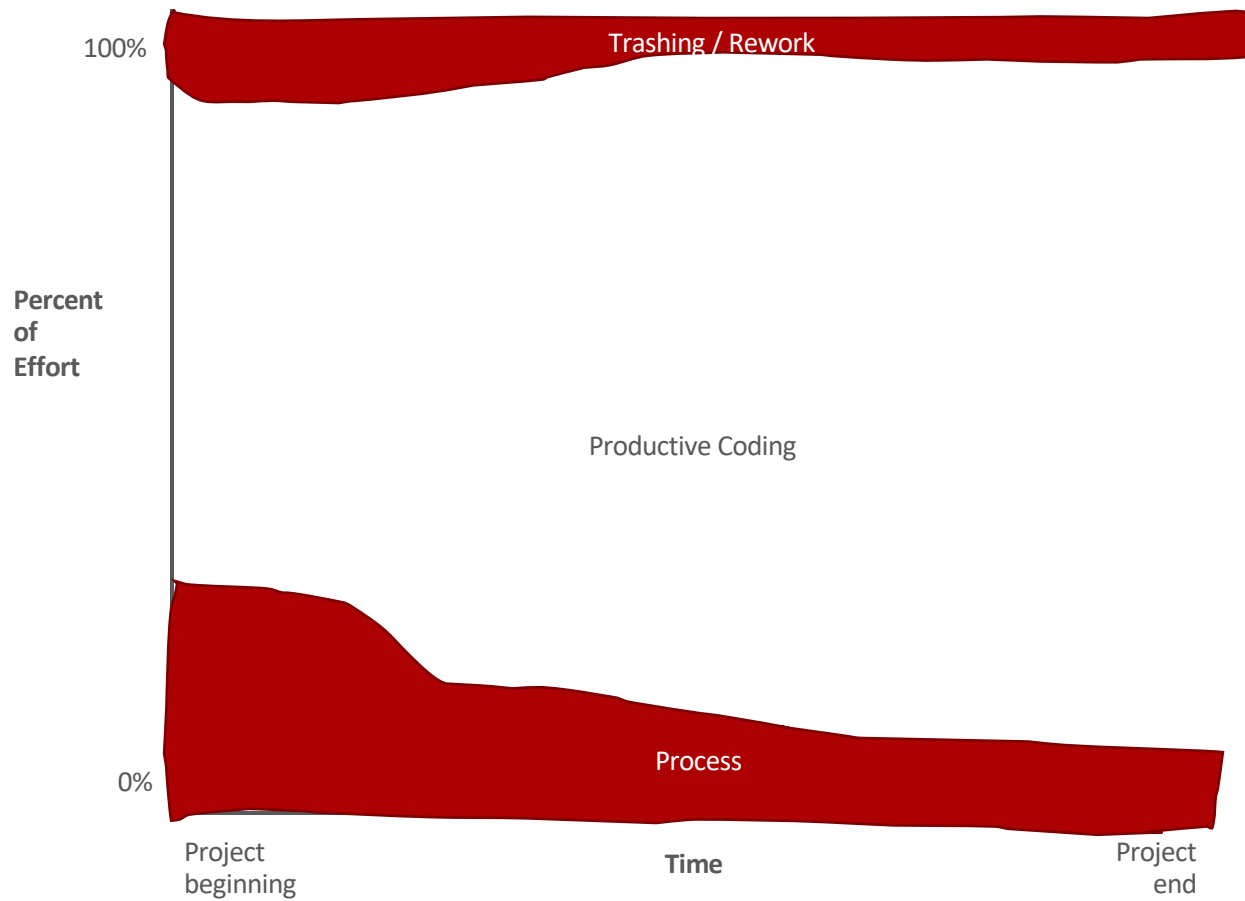


Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller tasks and schedule and monitor them
- Planning and conducting quality assurance
- Have daily status meetings
- Use Docker containers to push code between developers and operation

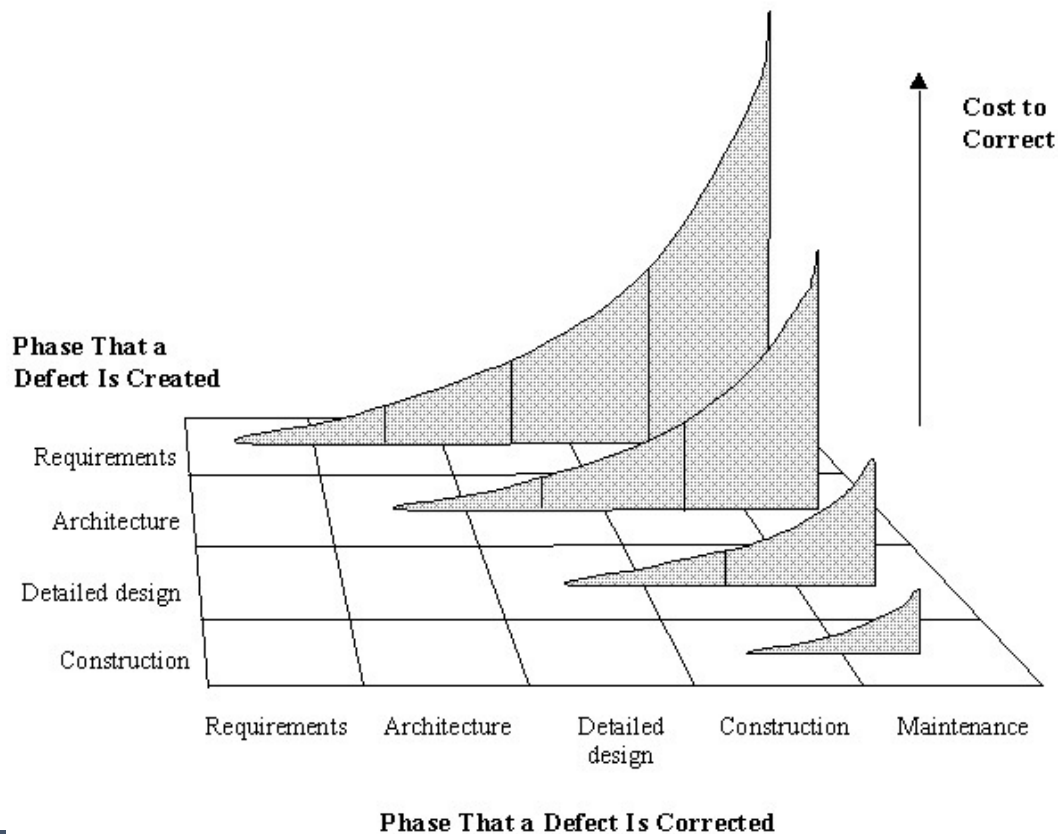
Example process issues

- Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
- Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- Defect Tracking: Bug reports collected informally, forgotten
- System Integration: Integration of independently developed components at the very end of the project. Interfaces out of sync.
- Source Code Control: Accidentally overwritten changes, lost work.
- Scheduling: When project is behind, developers are asked weekly for new estimates.



Hypothesis

- Process increases flexibility and efficiency
- Upfront investment for later greater returns



Planning

Estimating Effort

Task: Estimate Time

- A: Simple web version of the Monopoly boardgame with Pittsburgh street names
 - Team: just you
- B: Bank smartphone app
 - Team: you with team of 4 developers, one experienced with iPhone apps, one with background in security
- Estimate in 8h days (20 work days in a month, 220 per year)

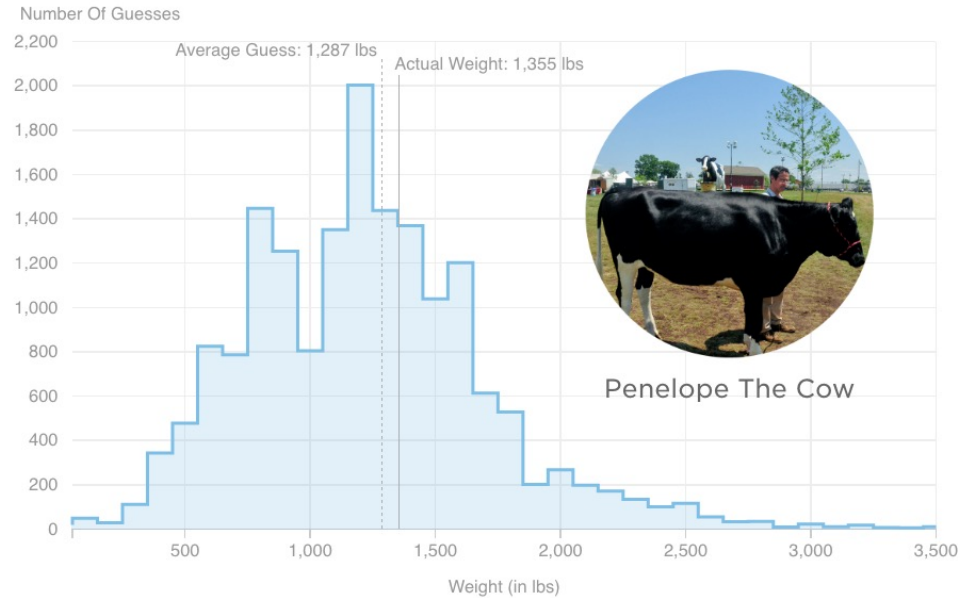
Revise Time Estimate

- Do you have comparable experience to base an estimate off of?
- How much design do you need for each task?
- Break down the task into ~5 smaller tasks and estimate them.
- Revise your overall estimate if necessary

π

How Much Does This Cow Weigh?

(All People)



Source: The Internet.

Credit: Quoc Trung Bui/NPR



XS



S



M



L



XL

made by **:codica**

codica.com

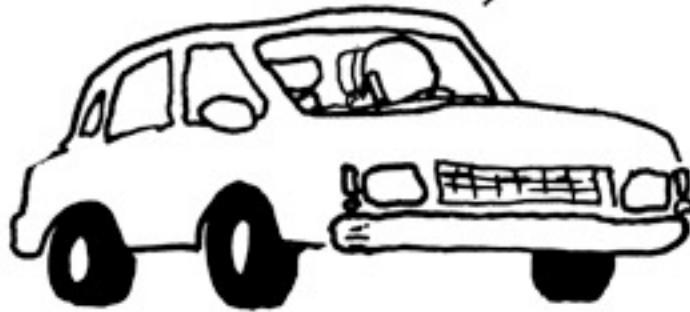
Measuring Progress?

- “I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”

I'M JUST OUTSIDE TOWN, SO I SHOULD
BE THERE IN FIFTEEN MINUTES.

ACTUALLY, IT'S LOOKING
MORE LIKE SIX DAYS.

NO, WAIT, THIRTY SECONDS.



THE AUTHOR OF THE WINDOWS FILE
COPY DIALOG VISITS SOME FRIENDS.

Measuring Progress?

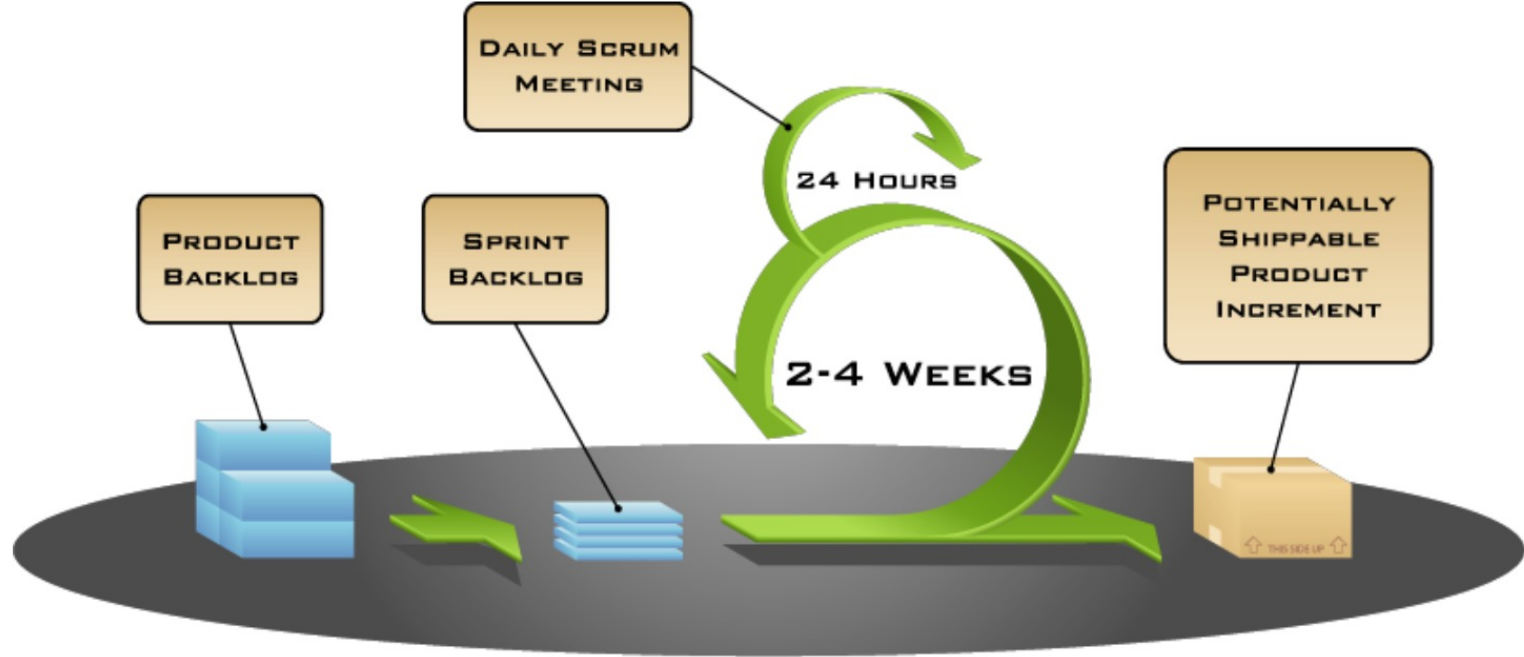
- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?



Milestones and deliverables

- Making progress observable, especially for software
- Milestone: clear end point of a (sub)tasks
 - For project manager
 - Reports, prototypes, completed subprojects
 - "80% done" not a suitable mile stone
- Deliverable: Result for customer
 - Similar to mile stone, but for customers
 - Reports, prototypes, completed subsystems

Brief intro to Scrum



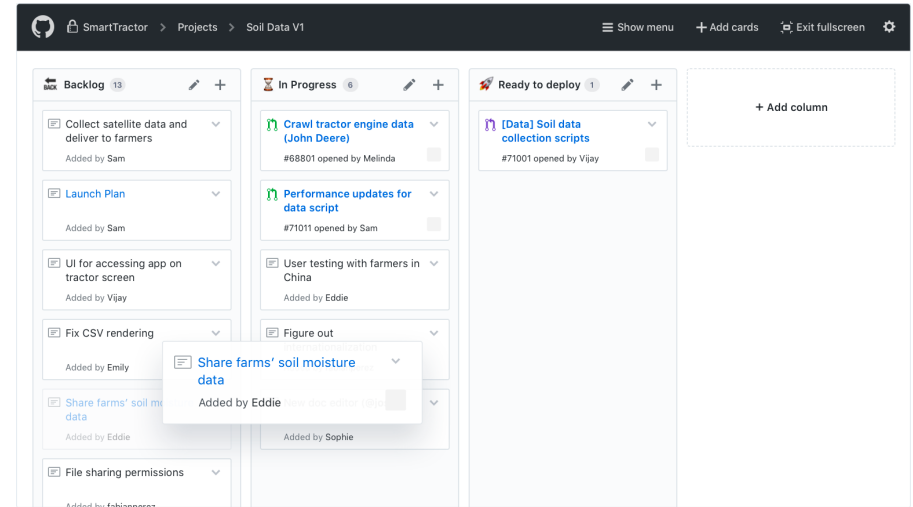
Elements of Scrum

- Products:
 - Product Backlog
 - Sprint Backlog
- Process:
 - Sprint Planning Meeting
 - Daily Scrum Meeting
 - Sprint Retrospective
 - Sprint Review Meeting

Product Backlog/Sprint Backlog

- The product backlog is all the features for the product
- The sprint backlog is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:
 - Fine-grained
 - Estimated
 - Assigned to individual team members
 - Acceptance criteria should be defined
- User Stories are often used

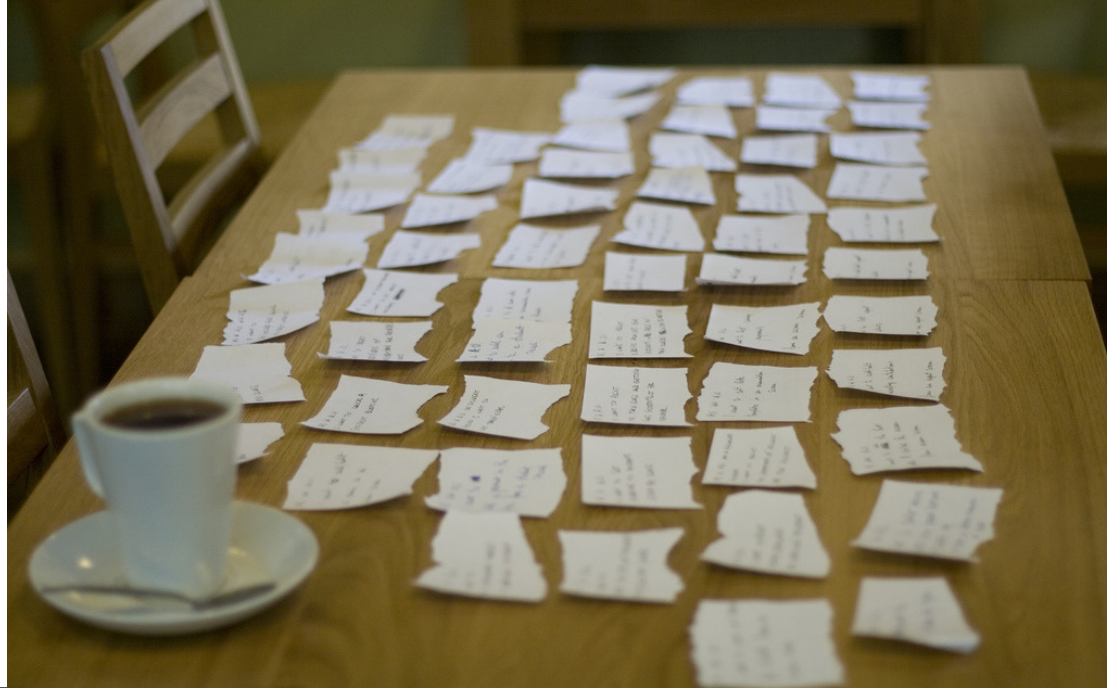
Backlog – information radiators



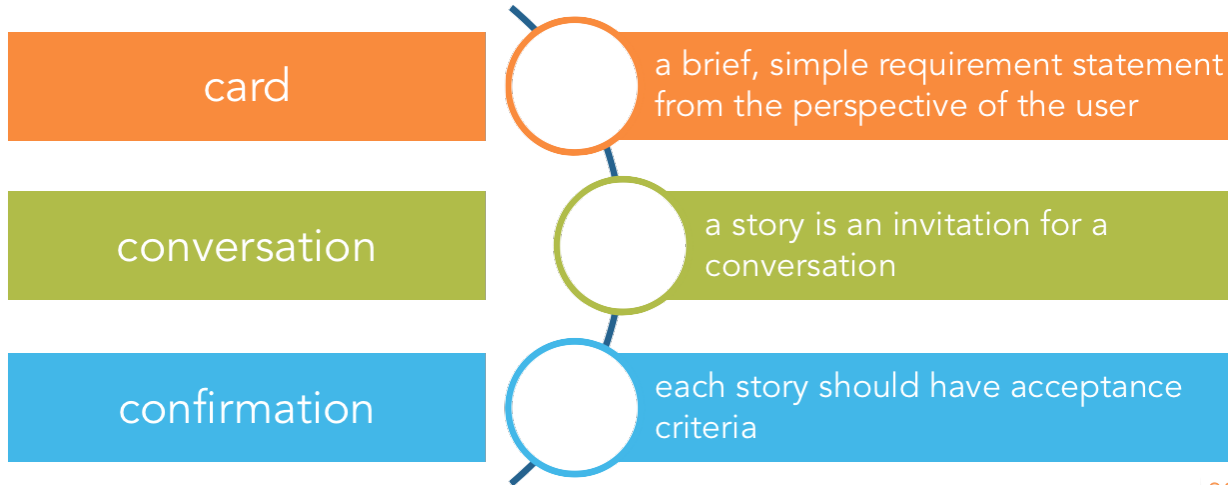
Scrum Meetings

- Sprint Planning Meeting
 - Entire Team decides together what to tackle for that sprint
- Daily Scrum Meeting
 - Quick Meeting to touch base on :
 - What have I done? What am I doing next? What am I stuck on/need help?
- Sprint Retrospective
 - Review sprint process
- Sprint Review Meeting
 - Review Product

User Stories



User Stories



one 80

The card

- “As a [role], I want [function], so that [value]”
- Should fit on a 3x5 card

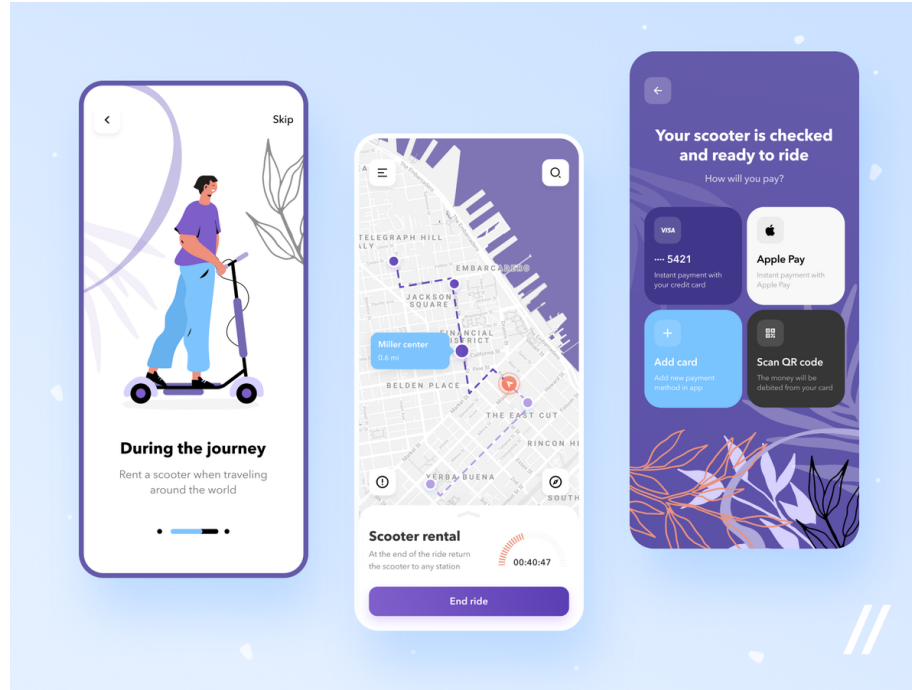
The conversation

- An open dialog between everyone working on the project and the client
- Split up Epic Stories if needed

The Confirmation

- A confirmation criterion that will show when the task is completed
- Could be automated or manual

Exercise



<https://dribbble.com/shots/12512417-Scooter-Rental-App-Design>

How to evaluate user story?

Follow the INVEST
guidelines for good
user stories!



Source: <http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/>

one | 80
SERVICES

Independent

- Schedule in any order.
- Not overlapping in concept
- Not always possible



Negotiable



- Details to be negotiated during development
- Good Story captures the essence, not the details

Valuable



- This story needs to have value to someone (hopefully the customer)
- Especially relevant to splitting up issues

Estimable



- Helps keep the size small
- Ensure we negotiated correctly
- “Plans are nothing, planning is everything” -Dwight D. Eisenhower

Small

- Fit on 3x5 card
- At most two person-weeks of work
- Too big == unable to estimate



Testable



- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test == do not understand

Activity

Follow the INVEST
guidelines for good
user stories!



one | 80
SERVICES



TEAMWORK (STUDENT TEAMS)

(MORE ON TEAMS IN REAL PROJECTS LATER IN THE COURSE)

Expectations

- Meet initially and then regularly
- Review team policy
- Divide work and integrate
- Establish a process
- **Set and document clear responsibilities and expectations**
 - Possible Roles: Coordinator, Scribe, Checker, Monitor
 - Rotate roles every assignment
- Every team member should understand the entire solution

Team Policies

- see document
- Make agreements explicit and transparent
- Most teams will encounter some problem

Dealing with problems

- Openly report even minor team issues in individual part of assignments
- In-class discussions and case studies
- Additional material throughout semester
- We will attend one team meeting

Planning and In-Team Communication

- Asana, Trello, Microsoft Project, ...
- Github Wiki, Google docs, ...
- Email, Slack, Facebook groups, ...

Next Up..

- HW1 due tonight
- Next week: Code Archeology
 - Pre-class quiz/survey on Canvas
- HW2 will be out by end of the week
 - Meet your teams!
 - Same teams for subsequent HWs