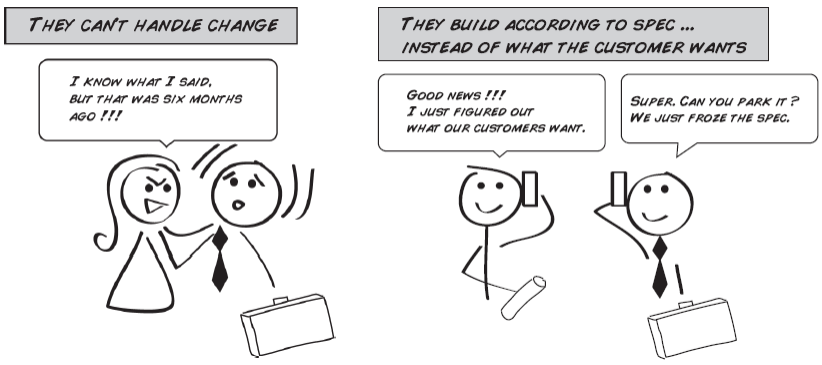
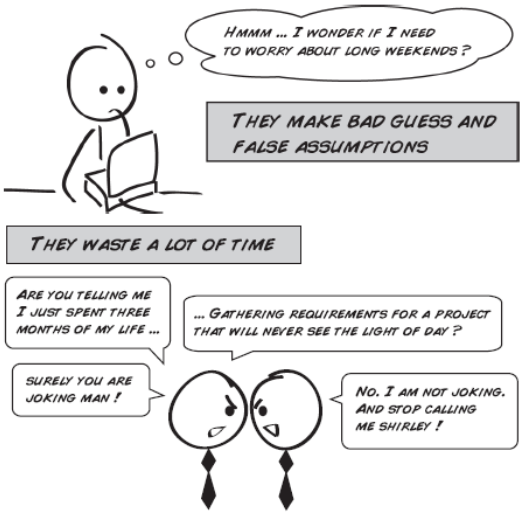
Agile Planning

The problem with documentation

- Argument: “Heavy” documentation too much for most business-style projects



Documentation



English Documentation

- Can be ambiguous (one reason for more formal documentation styles, e.g. ER diagram instead of English)
  - The man who hunts ducks out on weekends
  - Fat people eat accumulates
  - We painted the wall with cracks
  - "This agreement shall be effective from the date it is made and shall continue in force for a period of five (5) years from the date it is made, and thereafter for successive five (5) year terms, unless and until terminated by one year prior notice in writing by either party."

User Stories vs. Spec/Requirements

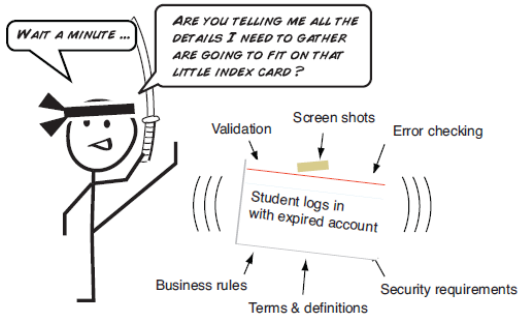
User stories 	Specifications & requirement docs 
Lean, accurate, just-in-time	Heavy, inaccurate, out-of-date
Encourage face-to-face communication	Encourage guesswork (false assumptions)
Simplified planning	Complex planning
Cheap, fast, easy to create	Expensive, slow, hard to create
Never out-of-date	Always out-of-date
Based on latest learnings	Based on little or no learning
Enable real-time feedback	Disable real-time feedback
Avoid false sense of accuracy	Promote false sense of accuracy
Allow for team-based collaboration and innovation	Discourage open collaboration and innovation

Agile principle

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agile User Stories

- Short descriptions of feature(s) the customer would like to see in their software
- Usually fits on an index card

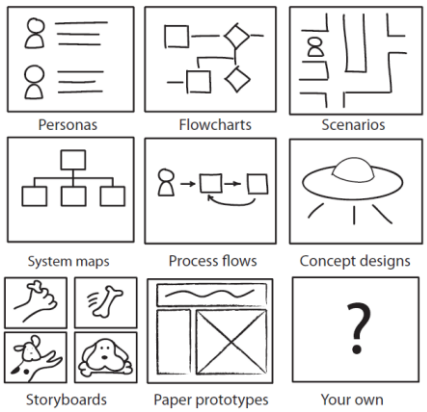


Elements of Good User Stories

- In language the customer understands
- Cuts end-to-end through layers of the architecture
- Independent of other user stories (as much as possible)
- Are negotiable, tradeoffs possible
- Are testable
- Are small and estimatable

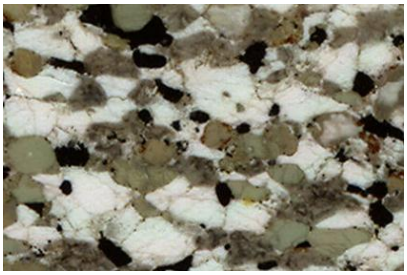
Extracting User Stories

- May need to do lots of brainstorming, draw lots of pictures



In-Class Exercise

- I am the client and all of you are the development team
- Help develop user stories for a thin section photomicrograph pixel counter



Check Stories

- Check INVEST
  - Independent, Negotiable, Valuable, Estimatable, Small, Testable
- Can something be re-cast as a constraint?
  - E.g. “Must be fast” to “Must load within 2 seconds”
- Scrub list, look for duplicates, consolidation or splitting of user stories

Analysis and Estimation

- You now have a stack of user stories
- Identify stories that require clarification
- Next we want to estimate how long they will take

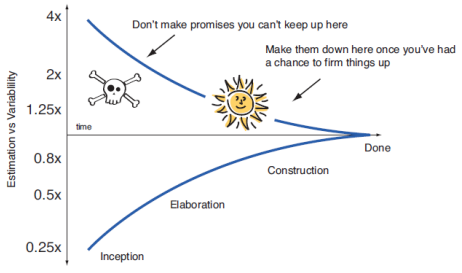
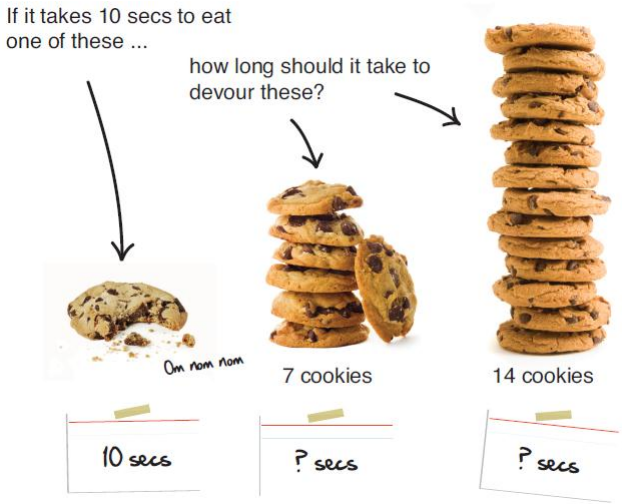


Figure 7.1: The cone of uncertainty reminds us of how greatly our estimates can vary at different stages throughout the project.

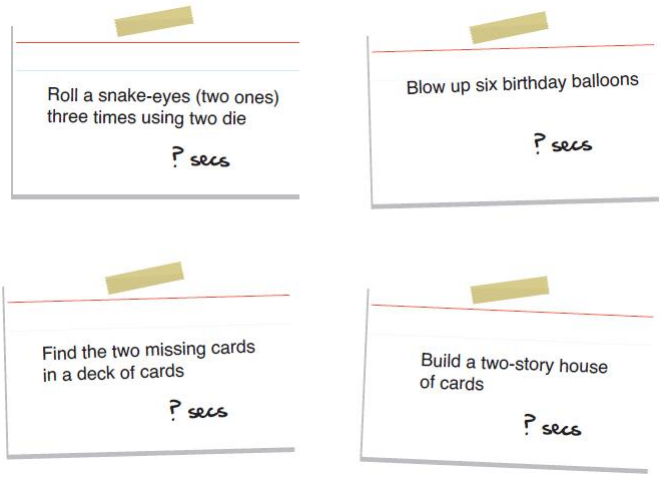
Relative Estimation

- Estimate coding time required for each story, but not in actual time, but in "units".
  - Joshua Kerievsky uses NUTs: Nebulous Units of Time
  - Idea is to convey the relative sizes of stories
  - Tough to do because you don't know what units represent until a few iterations are done, but they will shape up as time goes on

Relative Estimation

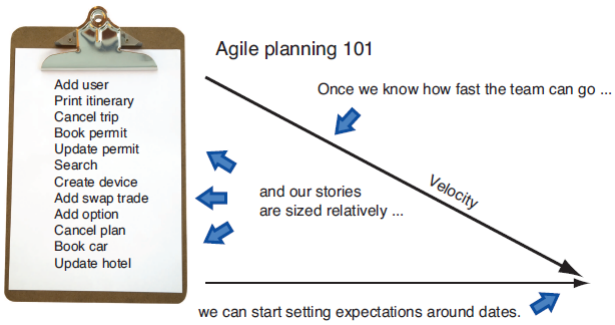


How Long?



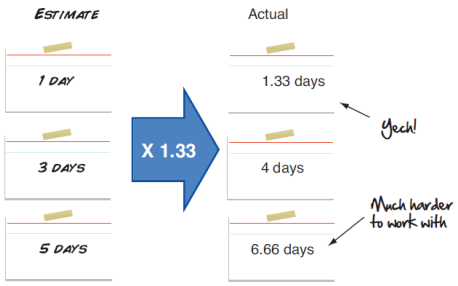
Estimation

- Humans are better at relative than absolute estimation
- Agile estimation is to size our stories relative to each other and keep track of time taken



Units of Time

- Say a story we estimated to take 3 days really took 4 days
- We could adjust actual calendar days to "programming days" by multiplying programming days by 1.333



Endless rejiggering?

False precision?

Point System

- Can avoid problems by using a point system
- Focus on relative sizes of the stories
  - Reminds us that estimates are guesses
  - Measure of pure size
  - Simple

Estimating Stories

- To estimate the user stories it may help to break them into tasks; discrete steps to complete the story
  - E.g. to save a document, you may have the task of creating the GUI to initiate the task, another task for the disk operation
- Brainstorm with your team for an estimate of units
- Tasks aren't shared with the client

In-Class Exercise

- Estimate units for thin section pixel counter user stories

Determining Workload

- You will need to convert from NUTS to actual time for an iteration
- Clients are initially not happy to get estimates in terms of units
  - Client: What's a unit?
  - Developers: We don't know.
  - Client: How many units can you do this week?
  - Developers: We don't know, but we can make an initial estimate, and it will get better every iteration and even within an iteration.
- If you estimated 20 NUTS the first iteration but you only completed 10 NUTS then you can generate a better estimate for the second iteration
  - Project spike useful here to get an initial estimate
  - Project velocity = NUTS completed / iteration

Estimating NUTs

- Estimate for the first iteration how many person-hours the group can collectively commit per week
  - Allow for time when you're not coding and not working
  - Make an estimate; the next one can be better
- Go to client and say how many units you can do per week
  - Consider how many people you have and how many hours each person can actually work
  - Sometimes you can make an estimate of units per hour
  - E.g. if 1 unit/hour and 20 person hours then you can do 20 units per week

Client Reevaluation

- Give client the user stories, estimates, and the total number of units you can do per week
- Client gets to pick the stories that add up to the total number of units
- Client doesn't get to add more stories beyond the total number of units
  - Important not to let the client get away with this, remind the client they can do different stories the next iteration
  - Have to prioritize and drop something if another being added

In-Class Exercise

- Estimate total hours, units per week for the thin section pixel counter project
- Client to prioritize

Dealing with Disappointment

- After a week perhaps you see your estimates weren't accurate
  - Usually programmers underestimate the time required
  - Reassess where you are with your group and immediately go to the client so he or she can determine how you should spend your remaining time
- Sometimes this is good news
  - If you only got to finish 10 units and you estimated 40, then you have better data for the next iteration
  - Estimates should get better each iteration; "surprises" are early, not later

Rinse and Repeat

- Even if you didn't complete as many stories as estimated the first iteration, the client should be happy with your honesty
- As the project progresses you should get better at knowing what you can do in an iteration
- Continue to keep the client informed and track where you are at all times
- Client may be unhappy the product is going slowly, but it's hard to argue with the data you are gathering and sharing

Communication

- Use BlackBoard wiki or forum to share information with your team members
- Good place to keep track of
  - Meeting notes
  - Issues or problems
  - Assigned tasks, estimates, actual time taken
    - Compare with actual time

Rules

1. The developers will be truthful in their estimates and the customers will believe these estimates
2. The developers will refine their estimates and the customers will refine their expectations based on the actual achievements in each iteration
3. During the iteration the developers will update the client as to the progress of the iteration. The client will use this information to quickly refine what is required in the current iteration.