Great Software Development (through iteration)

Pleasing the Customer

[Cartoon: Research shows that your best customers are creepy bearded guys. That same group also buys a high volume of potato chips and tissues. How's that help us? Two words: combo pack.]
Logistics Taking Shape

- [http://cseweb.ucsd.edu/classes/wi16/cse110-a](http://cseweb.ucsd.edu/classes/wi16/cse110-a)
- Email course staff: **cse-110-staff@googlegroups.com**

Labs and Project:
- Android-based: Please install Android SDK on laptop
- Form teams, ~6 people, must all attend same lab section
- Each team serves as both a customer team and a developer team
- As customer teams: Get an app built for you
- As developer team: Build an app for your customer team
- Agile methodology.
- Register teams:
  - [https://docs.google.com/spreadsheets/d/1D862EfXIIrreHPCm7JSpz06N1xEThpAe8uFLMZknYkQ/edit#gid=1740961395](https://docs.google.com/spreadsheets/d/1D862EfXIIrreHPCm7JSpz06N1xEThpAe8uFLMZknYkQ/edit#gid=1740961395)

- Weekly quizzes. First one Tuesday. More soon.
A Quick Background on (Software) Process

From improvisation to planning
Quality Control: A Short History

Quality control in early manufacturing was **Product-Centric** ("what")
- Regularly test product outputs
- Make adjustments to factory as needed
- But what to fix?

In mid 20\textsuperscript{th} century, shift to **Process-Centric** quality control ("how")
- Still test product outputs
- Also measure process elements
  - Plans, people, tools
- Use **cause-and-effect model** to adjust factory as needed
- Statistics to precisely track variation
- Buzzword: **Statistical Process Control**

- SE has inherited this legacy
- **SE methods are process-centric**
What’s a “Software Process” Anyway?

It’s the “how” that produces the “what” – quality software
- desired, on time, under budget

- A prescribed sequence of steps

- Steps include:
  - Planning
  - Execution
  - Measurement
    - Product, and process itself
    - Examples: bugs, progress, time, conflicts, re-work

- A software process is a self-aware algorithm
  - Observes and adapts according to measurements

- Agile processes are adaptive to the “customer”
  - Features, schedule, budget, priorities, markets, change
  - Must measure these as well as internal elements (correctness)
Venerable Model: Waterfall

- The "Waterfall"
  - Requirements Analysis
  - Design
    - Implementation
  - Verification
  - Operation/maintenance

- Good
  - Most expensive errors are higher up

- Bad
  - Doesn’t allow for change or discovery of errors along the way (Fatal flaw in many situations)
Iteration

The Heart of Agile Process
The Problem, the Solution

Big Bang (aka Waterfall)

You (based on initial conversation)

Customer (continues to think & biz continues to evolve after initial conversation)

Let’s travel the hypotenuse and straighten it out!

Let’s Iterate
Iterations are mini-projects

“Big Bang” Waterfall Development

Few well-defined quality measurement points, little data to guide process improvement

Quality measures can be performed more frequently, informing frequent adjustments
Planning with the customer

With customer, you work out:

1. List of requirements or features
2. Budget
3. Deadline

This triangle has a couple of names:
- Project management triangle
- Iron triangle

Conveys tradeoffs
Users change their minds, anyway, so stories need not be super precise. Instead, iterate to converge on timely and correct detail.
Take-Aways

Development Techniques

Iteration helps you stay on course

Plan out and balance your iterations when (not if) change occurs

Every iteration results in working software and gathers feedback from your customer every step of the way

Development Principles

Deliver software that’s needed

Deliver software on time

Deliver software on budget

Any concerns about this?
Requirements

Why can't the customer and developers get the requirements right in the first place?

A. Customer doesn’t know exactly what they want (what’s possible?)

B. Customer’s needs might change over time (more data – deliver iterations, eg)

C. Customer and developers have different points of view (miscommunication – different worlds)

D. Dev misunderstands customer (miscommunication)
Why does a user requirement have not just a definition, but also a **time estimate**, and a **customer priority**?

A. To see if we can meet the **deadline** (time is part of value)

B. Resources are limited (more eff. w/ resources, more profit, **prioritize for cost/benefit**)

C. To help you divide work into **iterations** (would be better if included priority)

D. Give pricing to the customer

*Iterations aren’t just buckets – they are in prioritized order.*
Iteration Length

How does a project team determine the length of its iterations?

A. Analyzing deadline and determining goals w/respect to that deadline

B. Proportionate to length of entire project (longer P → longer I)

C. Customer availability

D. Analyzing difficulty, risk, & value

The higher the risk (variance from bad to good scenarios), the more closely you want to monitor for frequent adjustments → shorter iterations
Productivity

Why doesn’t releasing to the customer all the time kill productivity?

A.  

B.  Key idea: Yes, there is overhead, but the insight from the frequent releases allows frequent process adjustments that keep you moving towards the goal (rather than away from it), thus avoiding costly re-work.

C.  

D.  

E.  
On Monday we talked about **Risk**
- Probability spread of **return on investment**
  - *investment*: time $\rightarrow$ *return*: get paid
- Minimize downside (don’t get paid)
- Maximize upside (customer happy, hires you again)

**Iteration addresses customer risks**
- Misunderstanding customer
- Customer changing mind
- Generally, disconnects with customer
Other project risks that iteration can address?

A. Here are just a few:

B.  
   • Technology risks (adopted tech fails)
   • Market risks (competitors)
   • Team risks – lack of skills, busy, infighting...

D. (you would be great at helping me come up with a bunch more.)
Take-Aways from Class Today

- Easy for break-down between customer & team
- Iteration – frequent releases – reduces this risk
- Customer communication is just one risk
  - Lots of other risks are addressed by iteration
- Iteration is example of general idea of making a large project act like a small one
  - Remember: real problem in development is scale