GATHERING REQUIREMENTS - I

Content from Chapter 3 of "Head First Software Development", Pilone et al.

Miami University Software Technology & Analysis Group (MUSTANG)
Computer Science & Software Engineering
Miami University, Oxford, Ohio, USA

AGENDA

- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

AGENDA

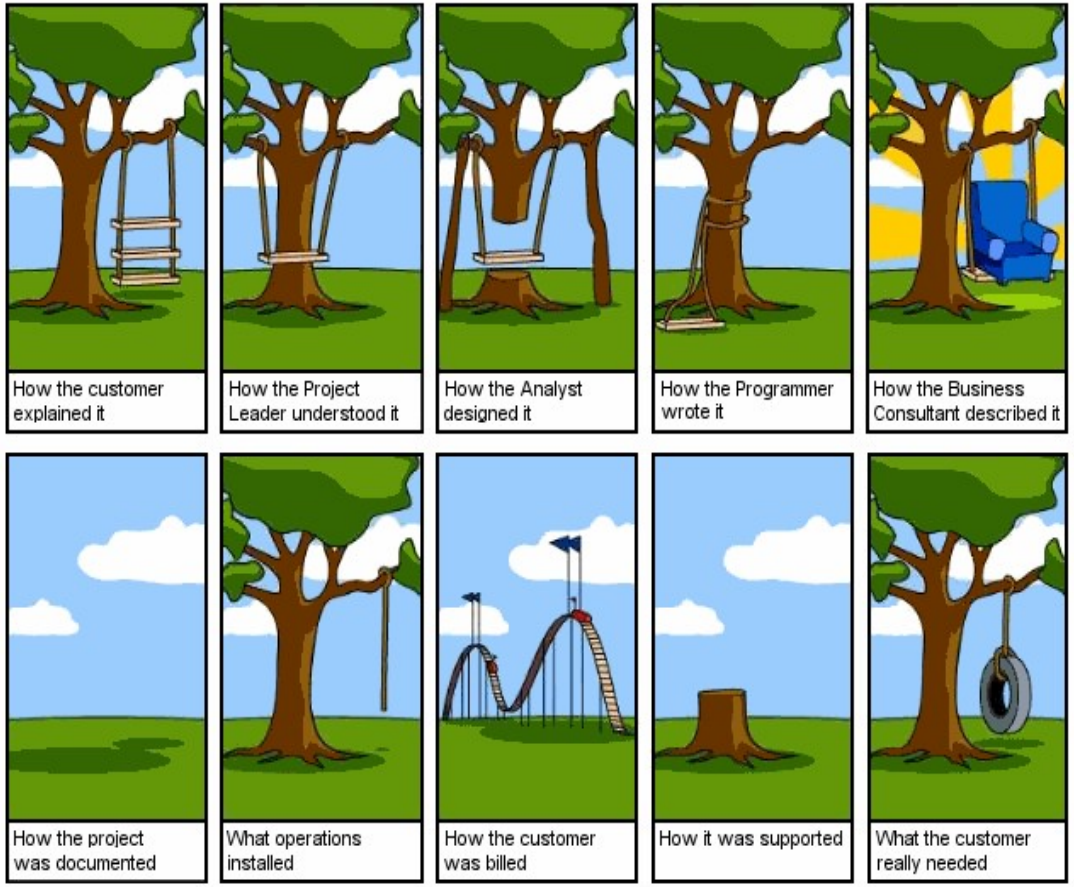
- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

REMEMBER

- The resources discussed in slides tagged [SWEBOK] and [GLOSSARY] refer to standardized documents

SOFTWARE DEVELOPMENT STAKEHOLDERS

- A Software Development Process has multiple "Process Actors" [SWEBOK]
  - Users
  - Customers
  - Software engineers



AGENDA

- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

SOFTWARE REQUIREMENTS

- Software Requirement:
  - "a software capability needed by a user to solve a problem..." [GLOSSARY]

GATHERING REQUIREMENTS

- We even iterate on the requirements

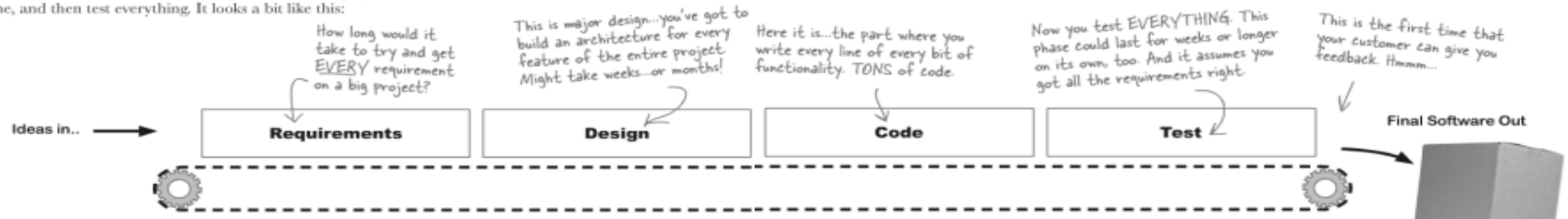
- Did I say more iteration? I meant to.

REQUIREMENTS ARE ITERATED

Each iteration is a mini-project

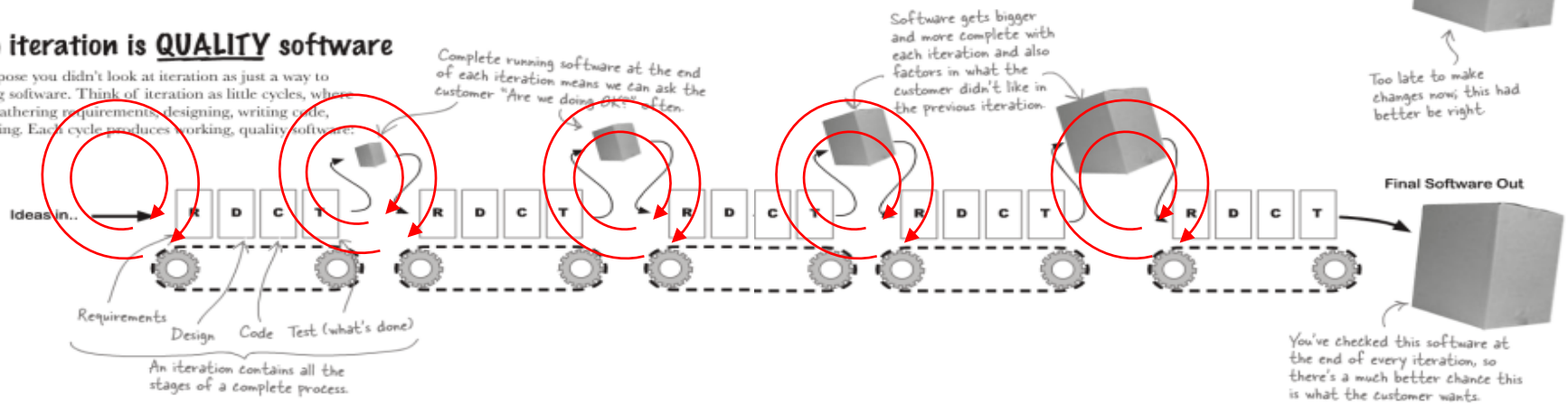
With iteration, you take the steps you'd follow to build the entire project, and put those steps into **each iteration**. In fact, each iteration is a mini-project, with its own requirements, design, coding, testing, etc., built right in. So you're not showing your customer junk... you're showing them well-developed bits of the final software.

Think about how most software is developed: You gather requirements (what your customer wants), build a design for the entire project, code for a long time, and then test everything. It looks a bit like this:



Each iteration is QUALITY software

But suppose you didn't look at iteration as just a way to write big software. Think of iteration as little cycles, where you're gathering requirements, designing, writing code, and testing. Each cycle produces working, quality software.

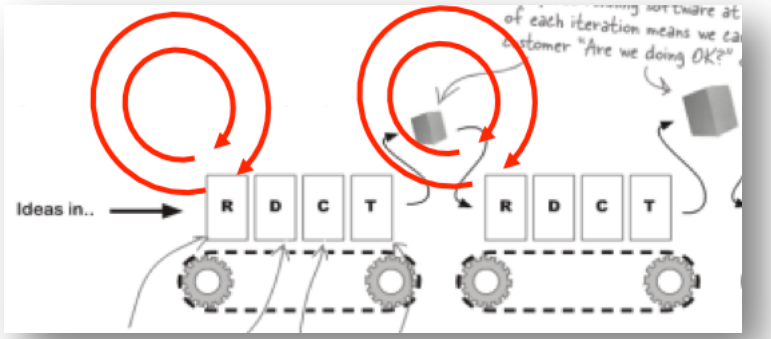


NOT THE SAME AS PHASE ITERATIONS

Iterate from ill-defined to defined

Rather than a new set of requirements each iteration

Repeat on same items until you get them right

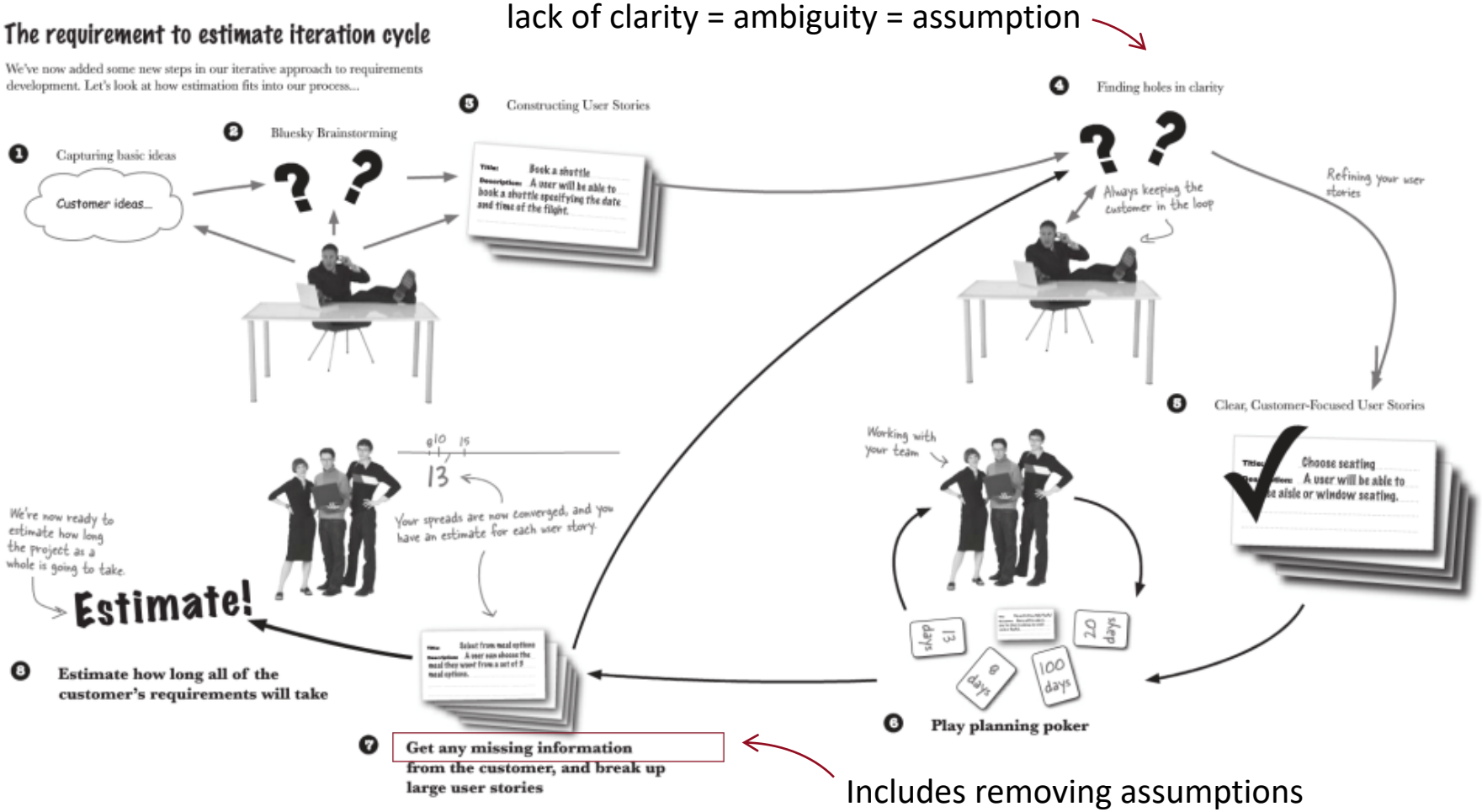


Goals of Requirements Gathering

- All the requirements
- Well understood
- Time estimates that you're confident in
- No Assumptions
  - Same as *ambiguity* here

THE REQUIREMENTS CYCLE (PROCESS, ALGORITHM)

lack of clarity = ambiguity = assumption



Includes removing assumptions

AGENDA

- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

REQUIREMENTS ELICITATION

- Requirements elicitation: “the process through which the acquirers (customers or users) and the suppliers (contractor) of a system discover, review, articulate, understand, and document the users’ needs and the constraints on the system and the development activity” [GLOSSARY]

REQUIREMENTS ELICITATION

- A process to **A,B,C,D,E** the needs and constraints of a system
  A. Discover
  B. Review
  C. Articulate
  D. Understand
  E. Document

Requirements Elicitation

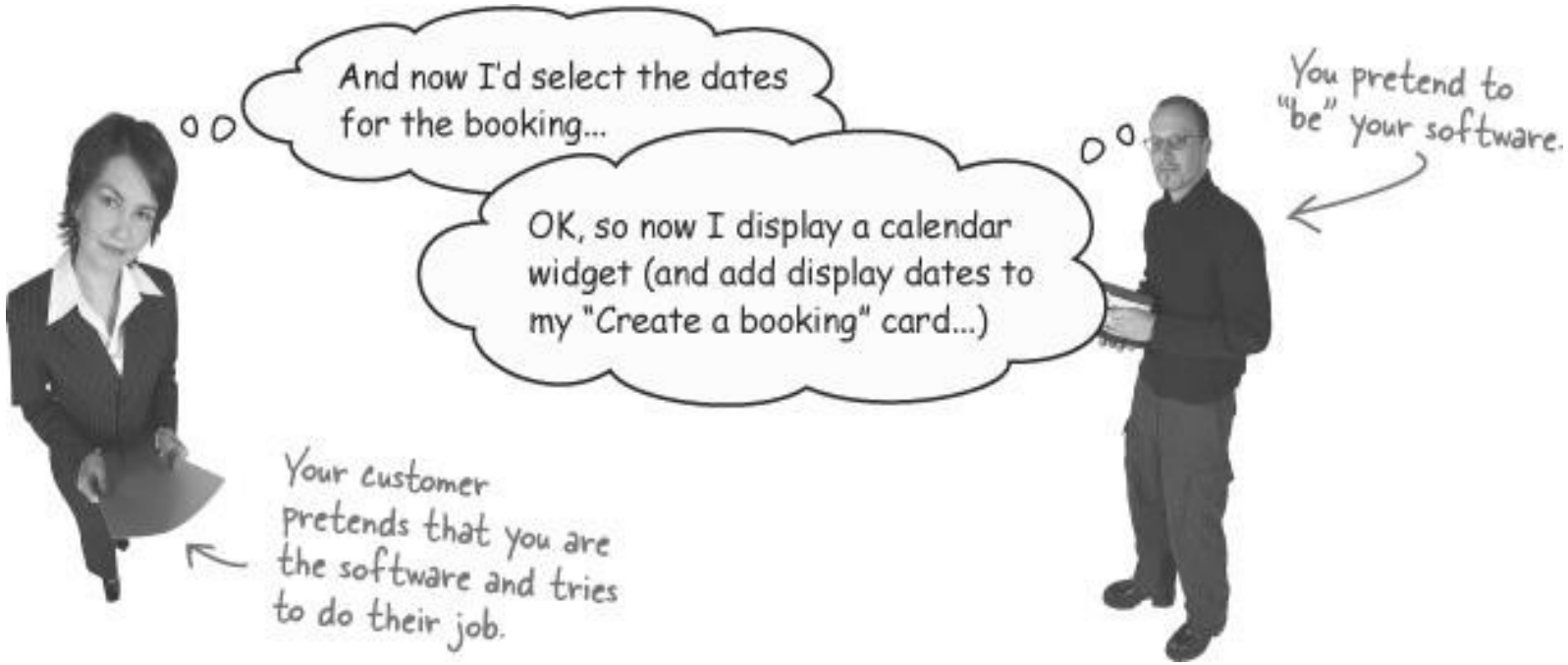
- Repeatedly ask good questions
- BlueSky with your customer
  - "sky is the limit"
  - stay open minded

Requirements Elicitation

- Role play
  - Have the customer play the software and have a dialog
- Observation
  - Watch how the client performs tasks without the software

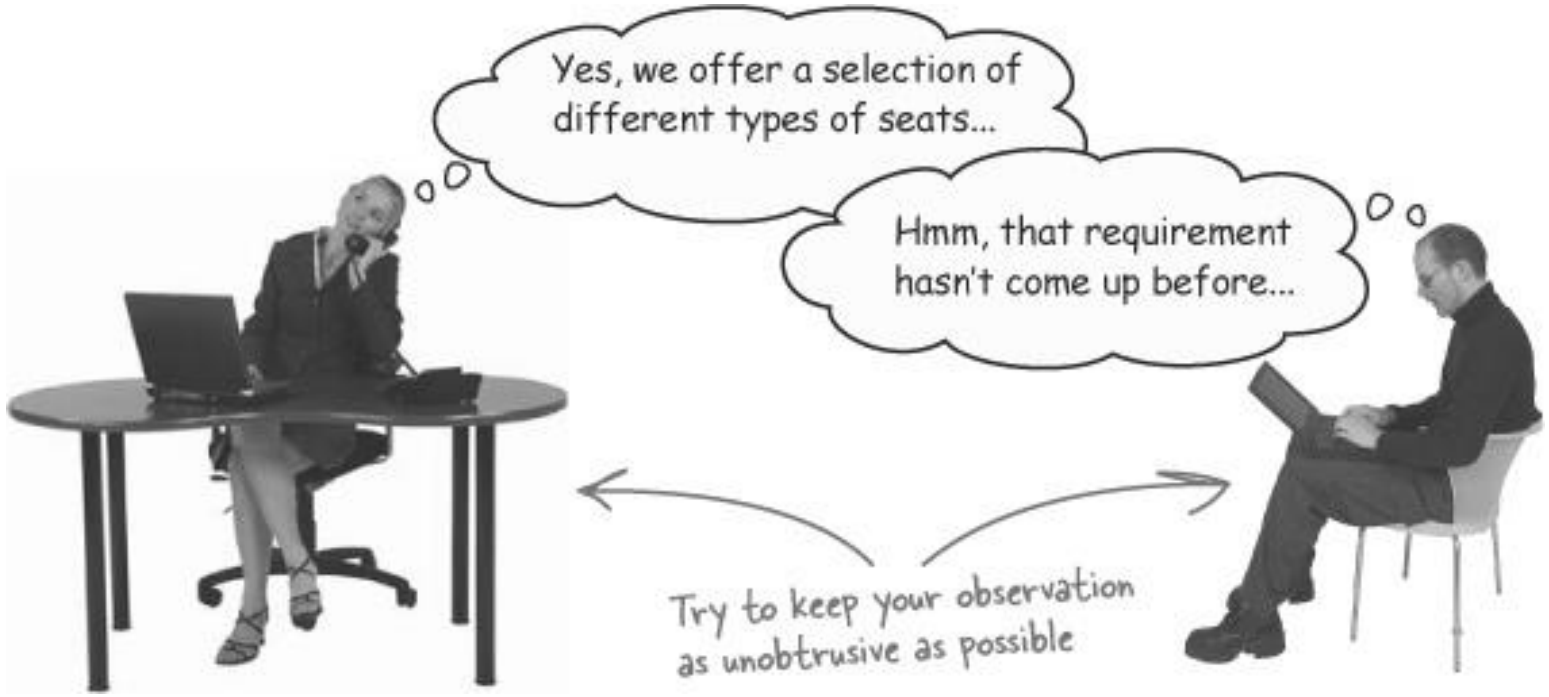
ROLE PLAY

- Role play
  - Have the customer play the software and have a dialog



OBSERVATION

- Observation
  - Watch how the client performs tasks without the software



Requirements Elicitation

- The customer should be heavily involved
  - Be sure all stakeholders are part of the process
- Keep asking

AGENDA

- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

REQUIREMENT ATTRIBUTES

- A requirement may have multiple attributes including:
  - Behavior (required)
  - Priority
  - Status
  - Time Estimate

REQUIREMENT: BEHAVIOR

- One thing the software should do
  - Must be verifiable!
- Written in language that the customer understand
  - ”User Stories" [TEXTBOOK]
- Be short

A nonfunctional constraint, but it is still captured as a user story

Title: Support 3000 concurrent users
Description: The traffic for Orion's Orbits is expected to reach 3000 users, all using the site at the same time.

Title: Pay with Visa/MC/PayPal
Description: Users will be able to pay for their bookings by credit card or PayPal.

Title: Review flight
Description: A user will be able to leave a review for a shuttle flight they have been on.

We've added to our cards from page 32 after the brainstorming with the customer.

Title: Order Flight DVD
Description: A user will be able to order a DVD of a flight they have been on.

Title: Order in-flight meals
Description: A user will be able to specify the meals and drinks they want during a flight.

Title: Book a shuttle
Description: A user will be able to book a shuttle specifying the date and time of the flight.

These were the requirements we came up with; yours could have been different.

Title: Choose seating
Description: A user will be able to choose aisle or window seating.

And we've added more detail where it was uncovered through brainstorming, role playing, or observation.

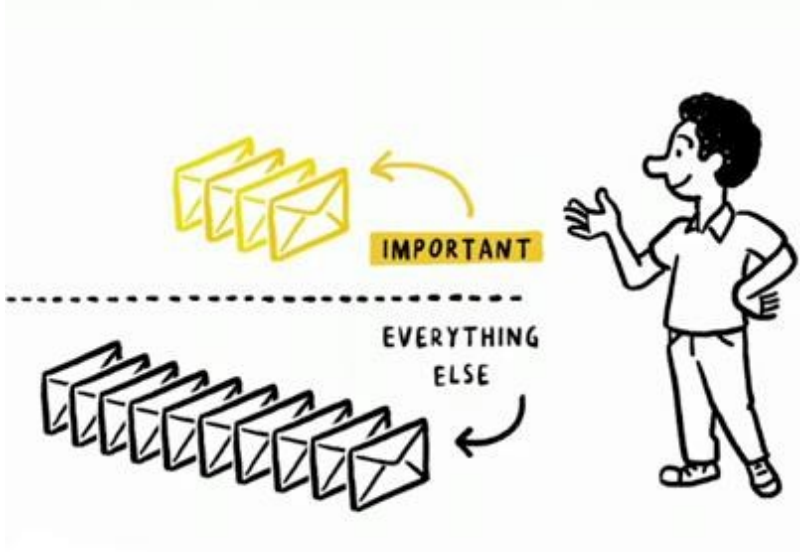
Title: Use Ajax for the UI
Description: The user interface will use Ajax technologies to provide a cool and slick online experience.

The boss isn't sure he understands what this requirement is all about.

REQUIREMENT: PRIORITY

- Relative to other requirements, how important is this requirement?

- Who sets the priority of a requirement?
  - A) User
  - B) Client
  - C) Software developer

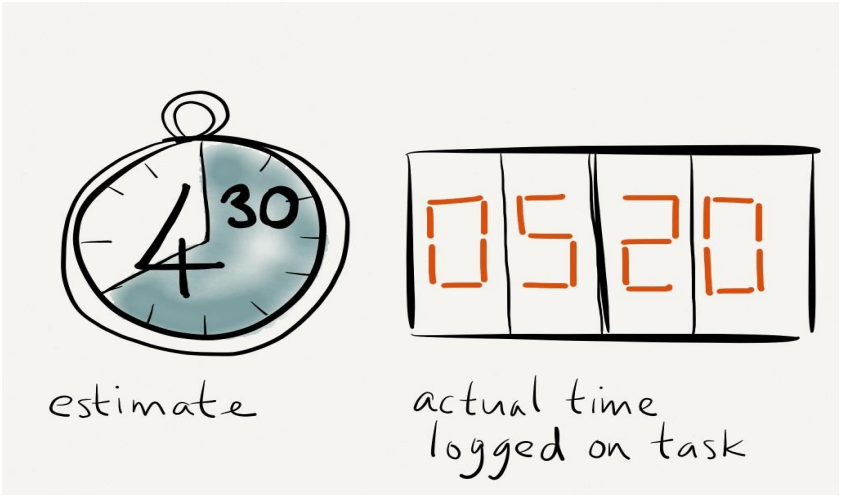


REQUIREMENT: STATUS

- What is the status of this user requirement
  - waiting?
  - in-progress?
  - updated time estimate?
  - complete?

REQUIREMENT: TIME ESTIMATE

- Ask all software developers to estimate how long it will take to complete a requirement
  - Days (TEXTBOOK prefers this)
  - Hours



REQUIREMENT: TIME ESTIMATE

- Among software developers (not the client) come as close as possible to a consensus
  - Large differences in estimates may indicate a problem
  - [TEXTBOOK] describes "planning poker" as an activity

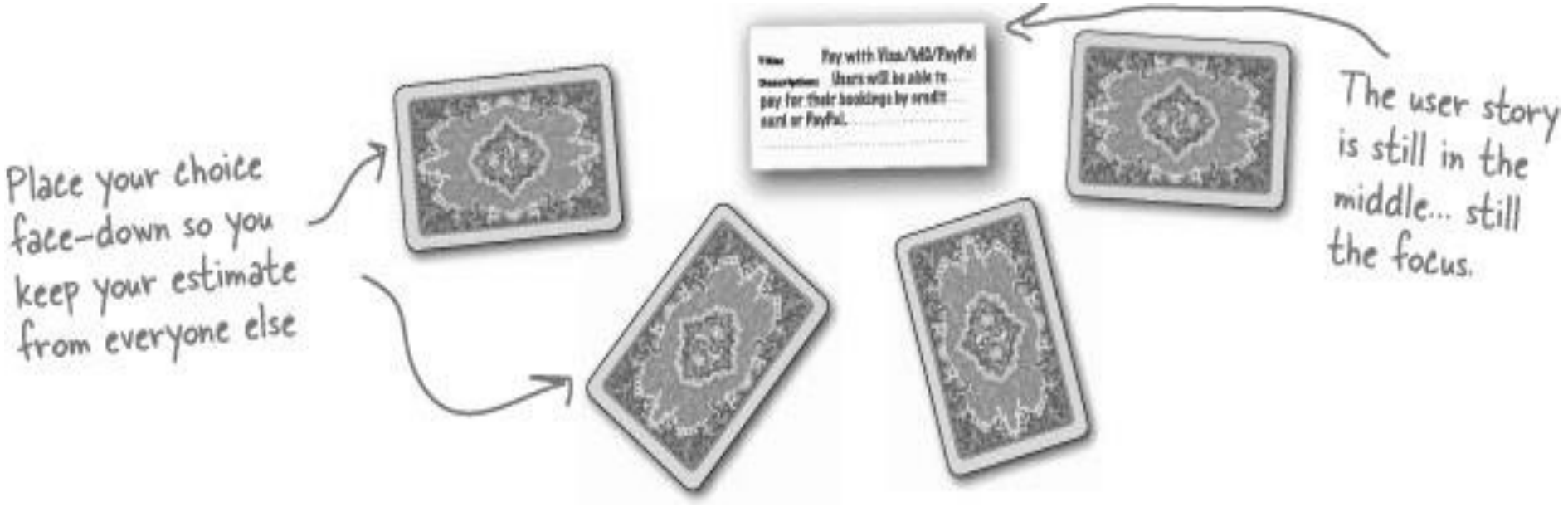
- Accuracy can be subverted by assumptions
  - Eliminate assumptions whenever possible
  - Surviving assumptions become **risks**

REQUIREMENT: TIME ESTIMATE

- Add the estimates for all requirements for the project estimate
  - What if the project estimate is too long?

PLANNING POKER

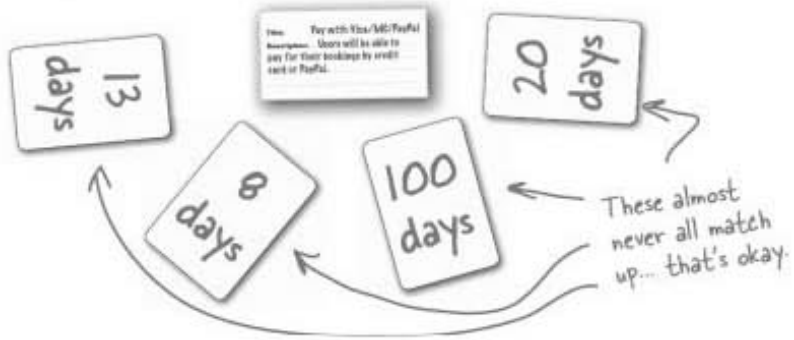
- Follow along with textbook -



Playing Planning Poker

1. Place a user story in the middle of the table
2. Everyone is given a deck of 13 cards.
   - Each card has an estimate written on one side
3. Everyone picks an estimate for the user story and places the corresponding card face down on the table
4. Everyone then turns over their cards at exactly the same time
5. The dealer marks down the spread across each of the estimates.

STEPS 4 & 5



5 **The dealer marks down the spread across each of the estimates.**

Whoever is running the game notes the spread across each of the estimates that are on the cards. Then you do a little analysis:

It's probably safe to figure an accurate estimate is somewhere in this range.

Ask the developer who played this card what they were thinking about; don't ignore them, try to pull out the assumptions they made.

The larger the difference between the estimates, the less confident you are in the estimate, and the more assumptions you need to root out.

REQUIREMENT ATTRIBUTES

- A requirement may have multiple attributes including:
  - Behavior - What does it do?
  - Priority - What order should we do it in?
  - Status - Is it done? started?
  - Time Estimate - How long will it take?

AGENDA

- Software Development Stakeholders
- Software Requirements
  - Requirements Elicitation
  - Requirement Attributes
  - Types of Requirements

Types of Requirements

- [SWEBOK]
  - Product vs Process
  - Functional vs Nonfunctional
  - Quantifiable
  - ~~System vs Software~~

Types of Requirements

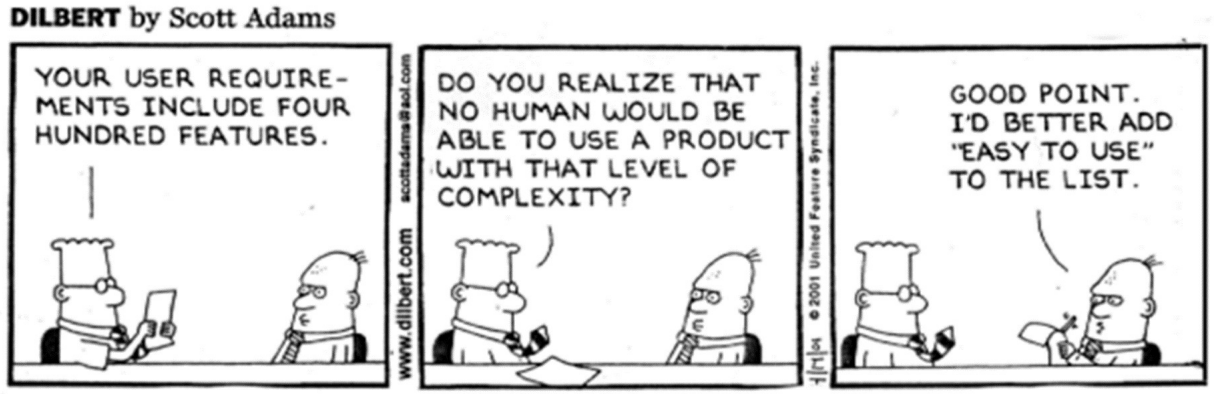
- [SWEBOK]
  - Product vs Process
  - Functional vs Nonfunctional
  - Quantifiable
  - System vs Software
  - Product: behavior of the software
    - "The software shall verify that a student meets all prerequisites before he or she registers for a course."
  - Process: constraints on the development of the software
    - "The software shall be written in Java"
    - "Software must pass a code security audit by ACME Security"

Types of Requirements

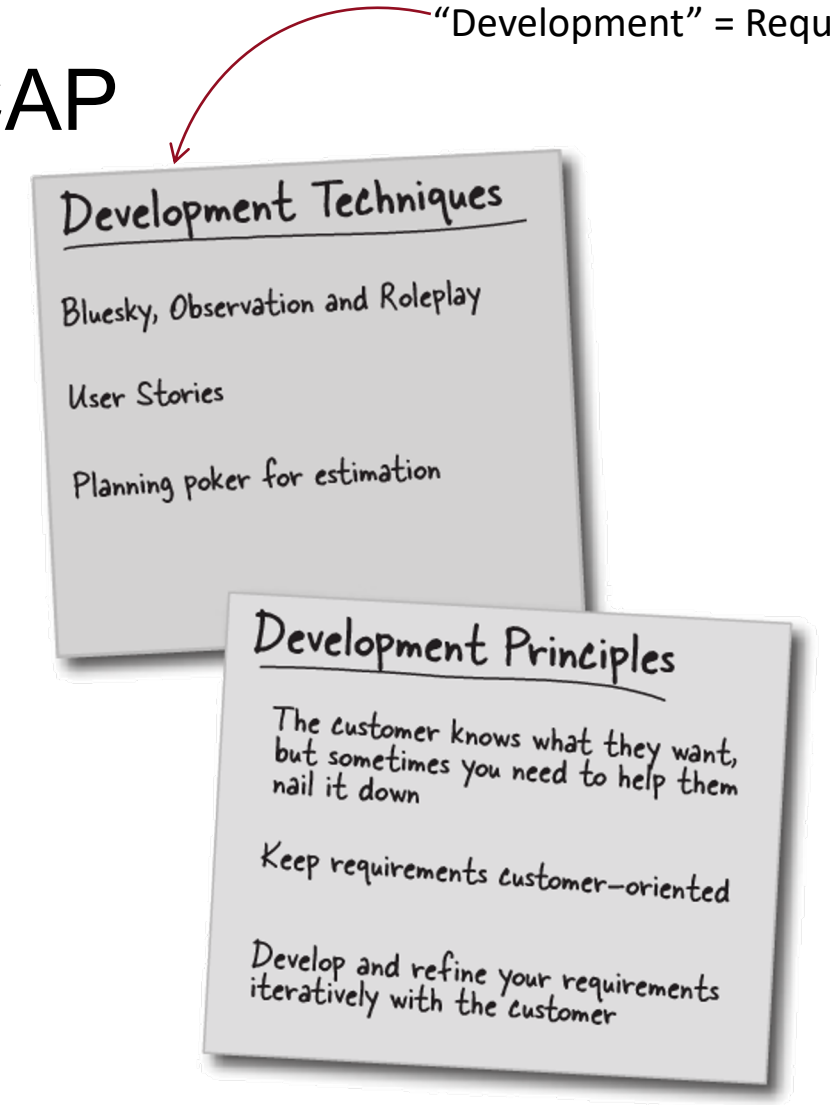
- [SWEBOK]
  - Product vs Process
  - Functional vs Nonfunctional
  - Quantifiable
  - System vs Software
  - Functional: functions that the software is to execute
    - "The software shall verify that a student meets all prerequisites before he or she registers for a course."
  - Nonfunctional: constrain the solution
    - "Verification of prerequisites shall not exceed 18 seconds"

Types of Requirements

- [SWEBOK]
  - Product vs Process
  - Functional vs Nonfunctional
  - Quantifiable
  - ~~System vs Software~~
  - Quantifiable
    - “software must increase the center’s throughput by 20%”
  - Quantifiable requirements are easier to verify!



RECAP



Requirements process/algorithm

1. Capture basic **ideas**
2. Bluesky **brainstorming**
3. Construct **user stories**
4. **Iterate on clarity** w/customer
5. **Refine** user stories
6. **Estimate** with planning poker
7. a. **Missing info** from customer
   b. **Test** your **assumptions**
   c. **Break up** large user stories
8. **Estimate all** requirements

NO ASSUMPTIONS

- Iterate with customer
- Test your assumptions

THE MOST IMPORTANT PROPERTY OF A REQUIREMENT

- Short and not too wordy
- Describes one thing
- Verifiable
- Assumptions are minimized

TAKE-AWAYS FROM CLASS TODAY

- Blueskying
  - ***Stakeholder = anyone affected by the project***
- Assumptions are a (hidden) form of:
  - Non-communication with customer
  - Technical uncertainty (skill, difficulty, …)
  - Risk
- **Iteration** is once again the solution
  - Iterate on user stories until assumptions are removed
  - Iterate with customer, selves, and in planning game (estimation)
- Good user stories are tricky, and so is estimation. Practice makes perfect!