Gathering Requirements

We even iterate on the requirements
Gathering Requirements

Did I say more iteration? I meant to.
Requirements are iterated
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
Avoiding this project nightmare
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
Removing Assumptions is Hard

- They’re invisible...
- Unspoken...
- Comforting...
- Effortless.

You should do this.

<table>
<thead>
<tr>
<th>Assumptions</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users will add profile photos</td>
<td>user study</td>
</tr>
<tr>
<td>Market is big enough</td>
<td>mkt research</td>
</tr>
<tr>
<td>Photo clustering is doable</td>
<td>hacking</td>
</tr>
<tr>
<td>Product page messaging is compelling</td>
<td>user study</td>
</tr>
</tbody>
</table>
Take-Aways

“Development” = Requirements

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

**Development Techniques**
- Bluesky, Observation and Roleplay
- User Stories
- Planning poker for estimation

**Development Principles**
- The customer knows what they want, but sometimes you need to help them nail it down
- Keep requirements customer-oriented
- Develop and refine your requirements iteratively with the customer
Blueskying

Why do we “Bluesky” with all stakeholders?

A. Experts at what they want
B. Unique points of view, think of different things
C. They’re paying
D. Reduces stakeholder assumptions
E. More people, more new ideas

Inclusive process avoids tensions and alienation.
Why don’t we **wait to estimate the user stories until the planning phase**? It would help us focus on just **what** the requirements are, which is hard enough.

A. Identify stories that are too big and split
B. Helps get rid of assumptions right away
C. Helps prioritize features, cost/benefit
D. All are right, but B is the most central
What’s wrong with this user story?

“The user will be able to click on an item and have javascript dynamically expand the item to show the item’s details.”

A. It mentions technical details (Javascript). User stories are written in customer language.

B. Because this user story has an “and” in it, you might think it could be broken into two user stories, but this is a colloquial use of “and” that means “causing”. This user story is not too big.
“myCity is a mobile app where you can see your myCity friends on a map, allowing you to message a nearby friend.”

3. Initial User Stories:
- Scroll & zoom the map
- Login
- filtering: by distance, groups
- visibility/access permissions
- Link account to facebook
- Status: class, driving, work
- Writing/receiving msgs – 2 stories

Commentary:
- Some of these read more like “blueskying” (step 2), such as connecting to FB. Great idea, but maybe out of scope. Could come out in clarification phase (step 4).
- Others are probably down the line (filtering)
- Others might be too detailed at this stage, but certainly might happen (zooming, login). They might come out in the refinement step (5), or during the planning phase as a task.
A couple of user stories

- Display map with user at the center (H)
- Show friends on the map (H)
- Click on friends, get a textbox, type & send msg (H)
- Map continues to track user’s changing location (M)
- Map updates with coming and going of friends (M)

Commentary: continuing to track user’s location might include a feature to allow setting the map view away from the user’s location (i.e., not tracking the user’s loc.)
“myCity is a mobile app where you can see your myCity friends on a map, allowing you to message a nearby friend.”

4. Finding Holes in Clarity:

- What’s nearby – city block to a mile
- Standalone, or leverage existing friend systems
  - GoogleTalk
- Message multiple people at once?
- Comm. with GoogleTalk substrate
- Birds-eye map or street-level
  - birds-eye, but really whatever Google/Android give you
- Android-only
- How do we deal with lots of buddies nearby?
- How show friends on map – icon, photo, etc.?

Commentary: these are awesome questions for removing ambiguity.
myCity requirements, continued

“myCity is a mobile app where you can see your myCity friends on a map, allowing you to message a nearby friend.”

5. Refine User Stories (refinements in bold):

- Display **birds-eye** map with user at the center (H)
  - Don’t worry about scalability issues right now (lots of buddies)
  - Show buddy handle on map

- Show **GoogleTalk** friends on the map (H)

- Click on friends, get a textbox, type & send msg (H)
  - No broadcast right now, but good idea for later
  - map and buddy list, don’t have to message from map

- Map continues to track user’s changing location, **with option to not track** (M)

- Map updates with coming and going of friends (M)
6. Play Planning Poker (1 user story)

Display birds-eye map with user at the center (H)

A. 2 person-days (2 people, 8 hours each)
B. 4 person-days
C. 6 person-days
D. 10 person-days
E. 16 person-days
myCity requirements: exercise at home

“myCity is a mobile app where you can see your myCity friends on a map, allowing you to message a nearby friend.”

7a-b. Get missing info, test assumptions:
myCity requirements: exercise at home

7c. Break up large user stories
8. Estimate all requirements
Take-Aways from Class Today

- **Blueskying**
  - Brings in stakeholders beyond customer
  - *Stakeholder = anyone affected by the project*
  - Both technical (miss something) and political (piss someone off)

- 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!