MIAMI UNIVERSITY

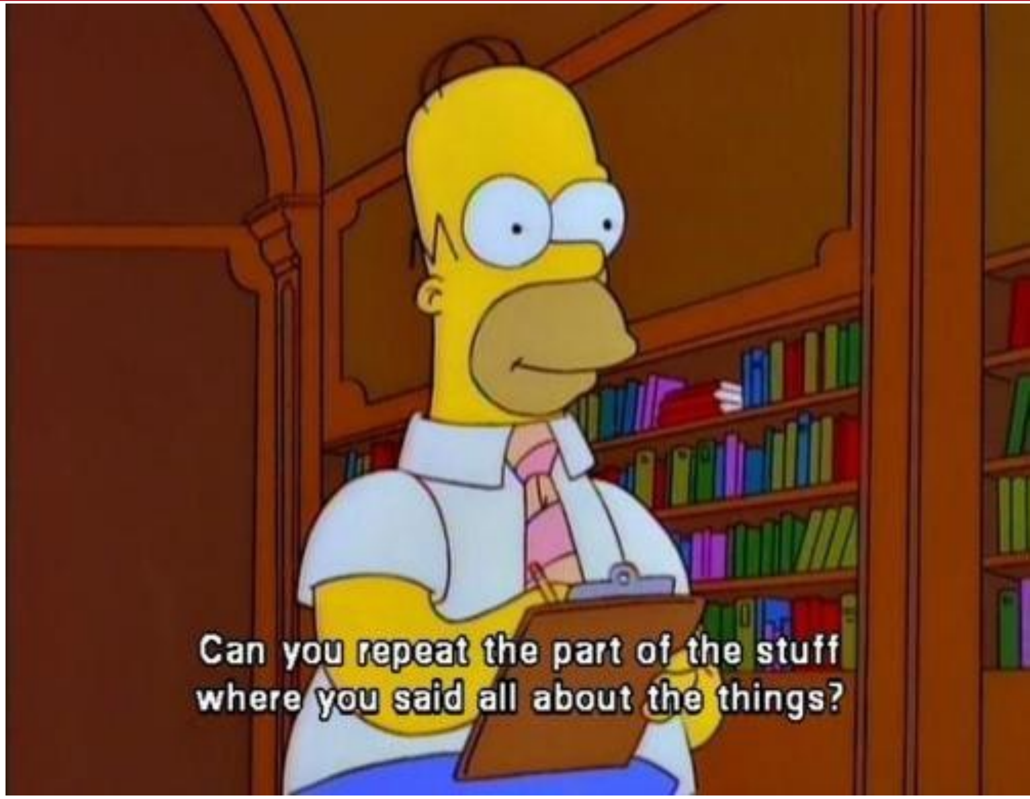
GATHERING REQUIREMENTS - II

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

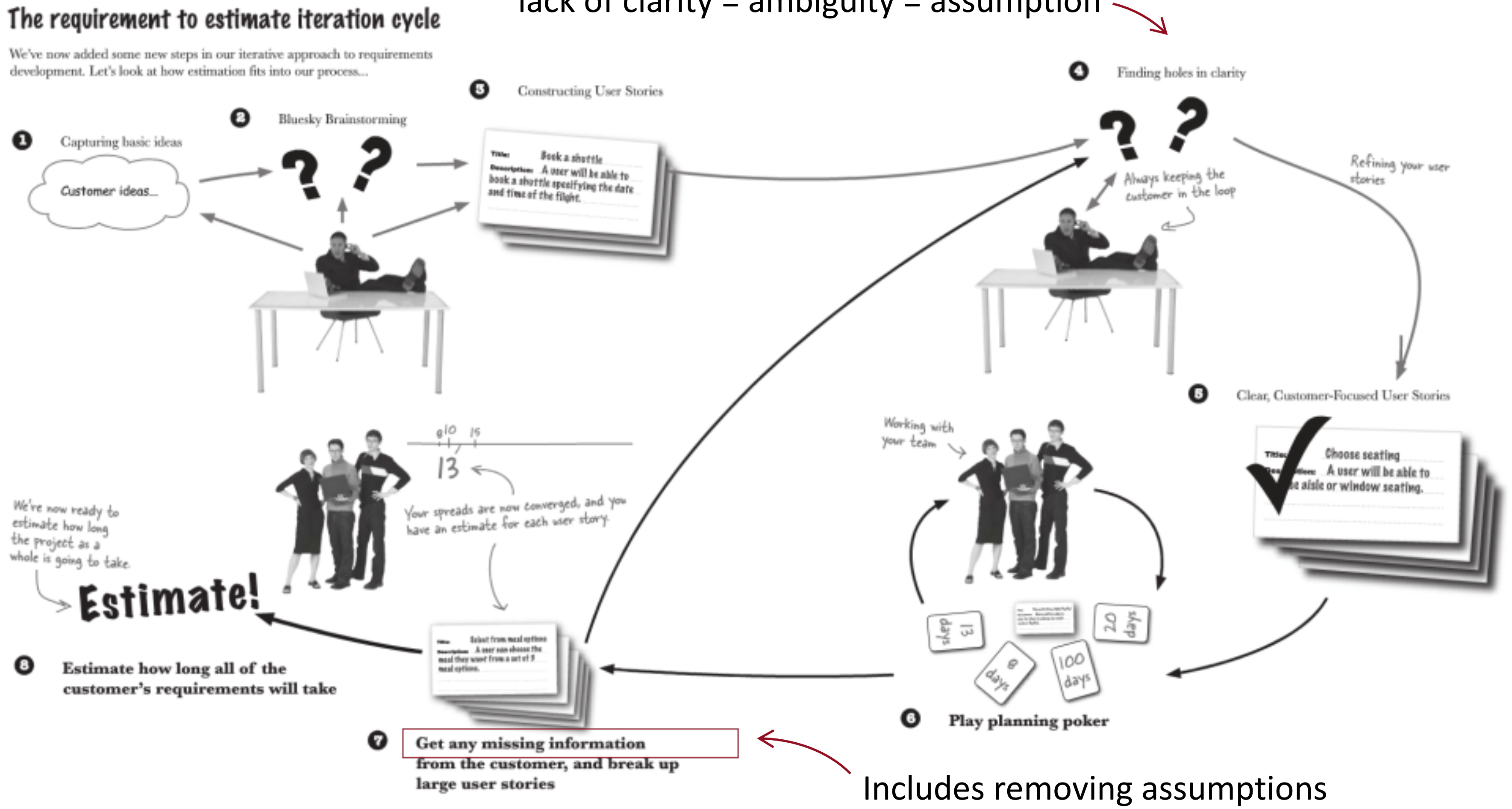
AGENDA

- Review
  - Case Study
- Finishing requirements


Can you repeat the part of the stuff where you said all about the things?

The Requirements Cycle (Process, Algorithm)

lack of clarity = ambiguity = assumption



The requirement to estimate iteration cycle

We've now added some new steps in our iterative approach to requirements development. Let's look at how estimation fits into our process...

1 Capturing basic ideas
Customer ideas...
2 Bluesky Brainstorming
3 Constructing User Stories
Title: Book a shuttle
Description: A user will be able to book a shuttle specifying the date and time of the flight.
4 Finding holes in clarity
Always keeping the customer in the loop
Refining your user stories
5 Clear, Customer-Focused User Stories
Title: Choose seating
Description: A user will be able to choose aisle or window seating.
Working with your team
6 Play planning poker
13 days
20 days
8 days
100 days
7 Get any missing information from the customer, and break up large user stories
Title: Select from meal options
Description: A user can choose the meal they want from a set of 3 meal options.
Your spreads are now converged, and you have an estimate for each user story.
13
We're now ready to estimate how long the project as a whole is going to take.
Estimate!
8 Estimate how long all of the customer's requirements will take

Includes removing assumptions

Case Study

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

REQUIREMENT TO USER STORIES: AMBIGUITY AND REFINEMENT

"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 might be too detailed at this stage, but certainly might happen (zooming, login). They might come out in the refinement step (**step 5**), or during the planning phase as a task.

A COUPLE OF USER STORIES

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

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

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."

4. Finding Holes in Clarity:

- What's nearby – city block to a mile
- 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.

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

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 (<u>refinements in bold</u>):

- Display **birds-eye** map with user at the center
  - **Don't worry about scalability issues right now (lots of buddies)**
  - **Show buddy handle on map**
- Show **GoogleTalk** friends on the map
- Click on friends, get a textbox, type & send msg.
  - **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**
- Map updates with coming and going of friends

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

MYCITY REQUIREMENTS, CONTINUED

6. Play Planning Poker (1 user story)

Display birds-eye map with user at the center

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

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

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

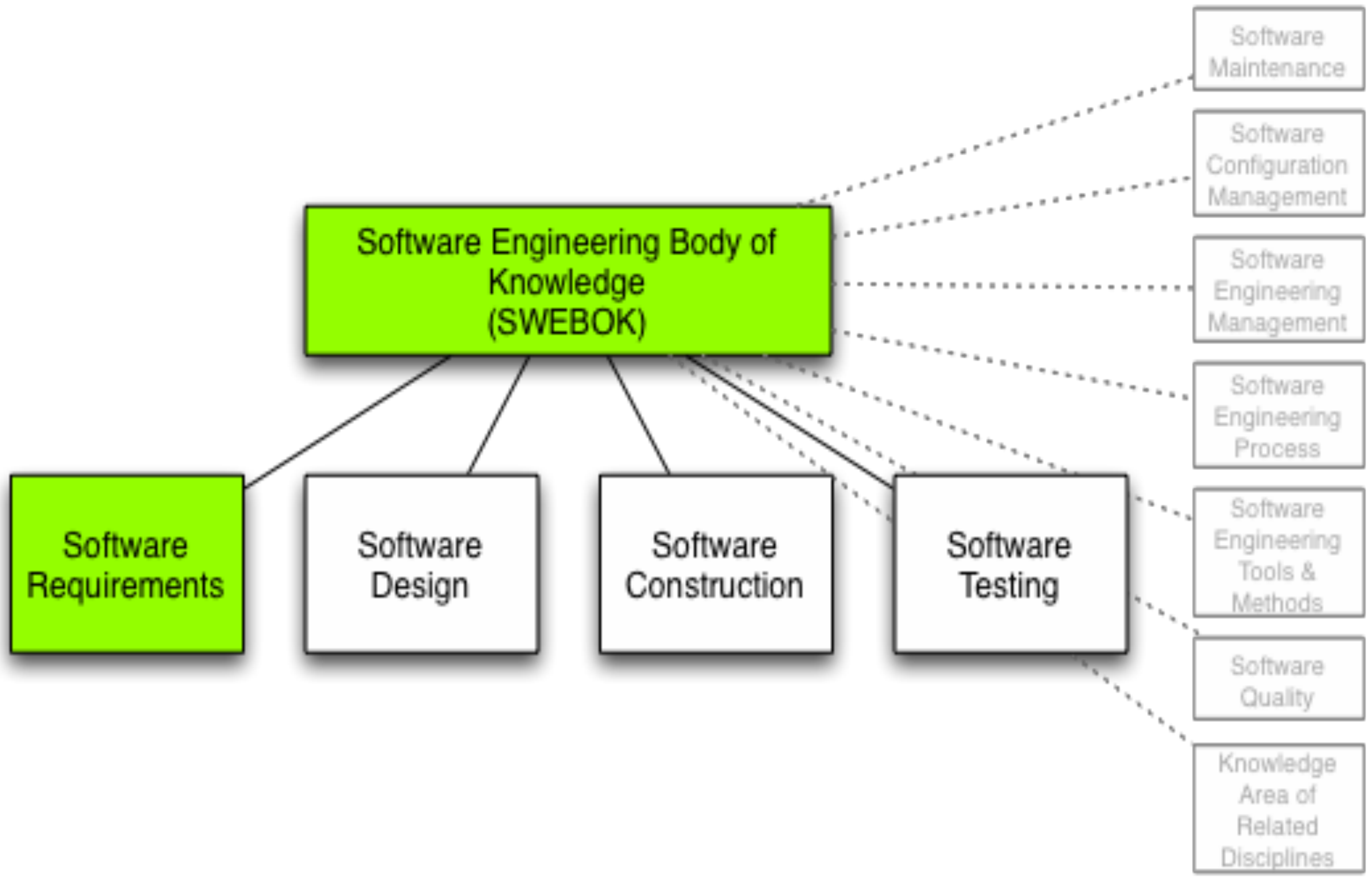
-

MYCITY REQUIREMENTS: EXERCISE AT HOME

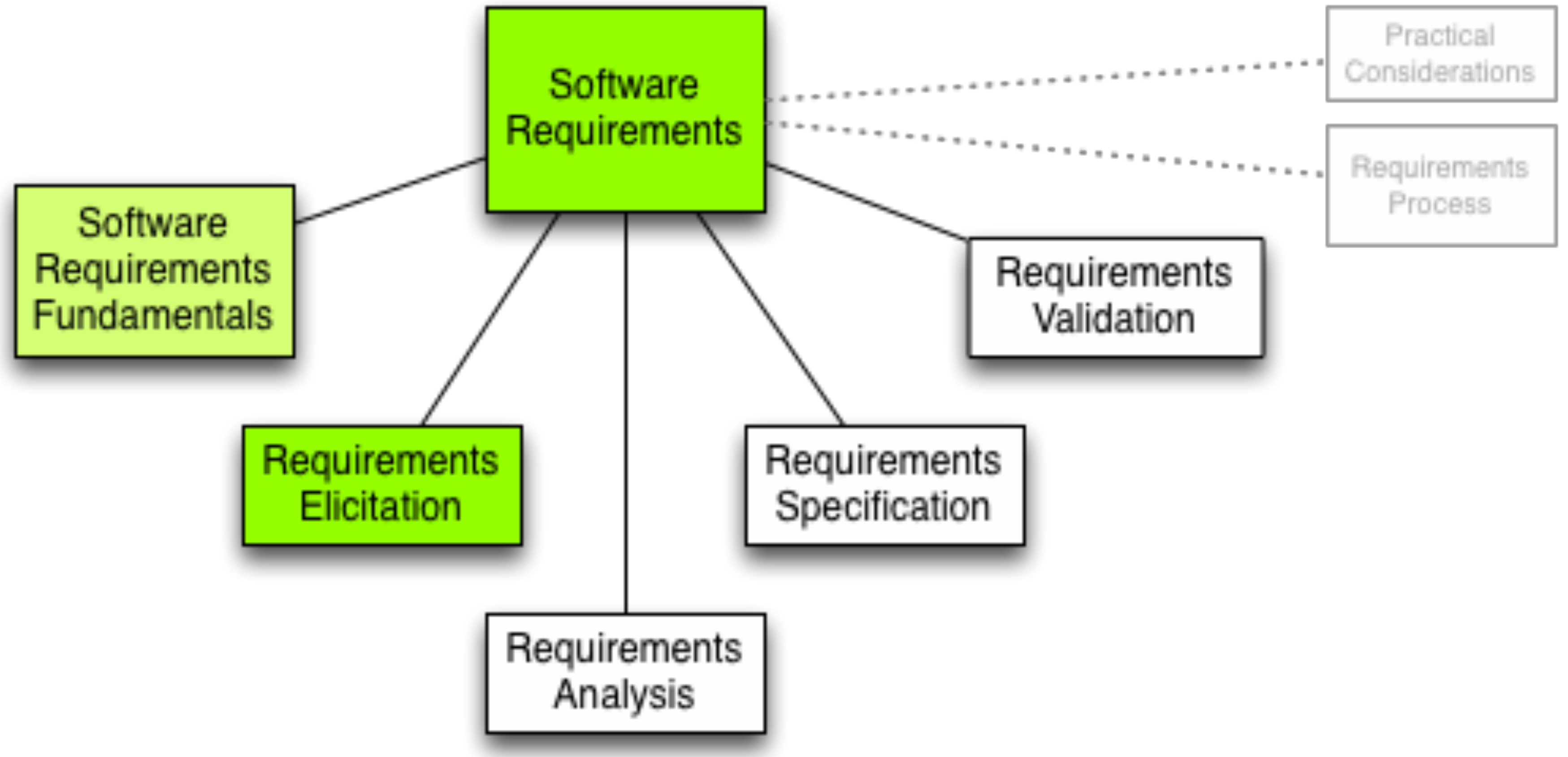
8. Estimate all requirements

▪

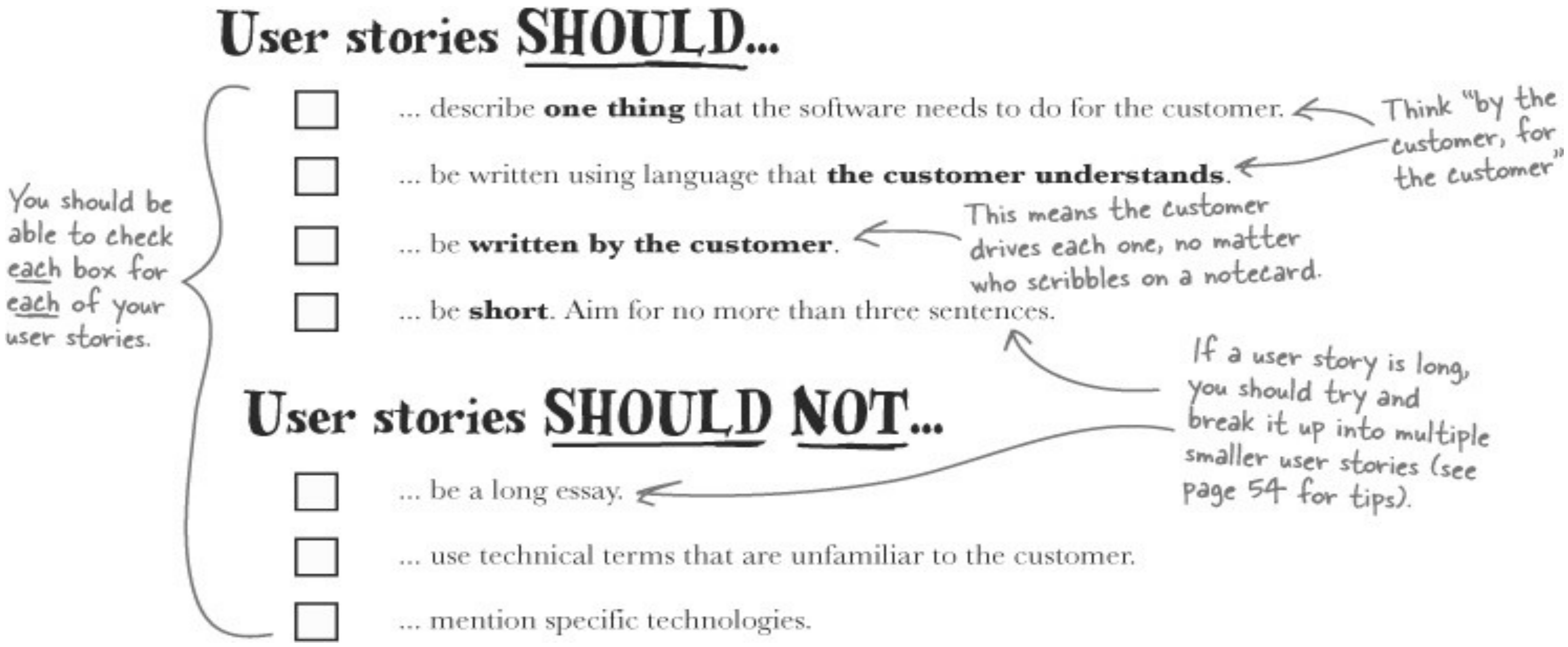
SWEBOK


Software Engineering Body of Knowledge (SWEBOK)
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Tools & Methods
Software Quality
Knowledge Area of Related Disciplines

SWEBOK


Software Requirements
Software Requirements Fundamentals
Requirements Elicitation
Requirements Analysis
Requirements Specification
Requirements Validation
Practical Considerations
Requirements Process

Requirements Must be Customer-Oriented



User stories SHOULD...

- [ ] ... describe **one thing** that the software needs to do for the customer.
- [ ] ... be written using language that **the customer understands**.
- [ ] ... be **written by the customer**.
- [ ] ... be **short**. Aim for no more than three sentences.

User stories SHOULD NOT...

- [ ] ... be a long essay.
- [ ] ... use technical terms that are unfamiliar to the customer.
- [ ] ... mention specific technologies.

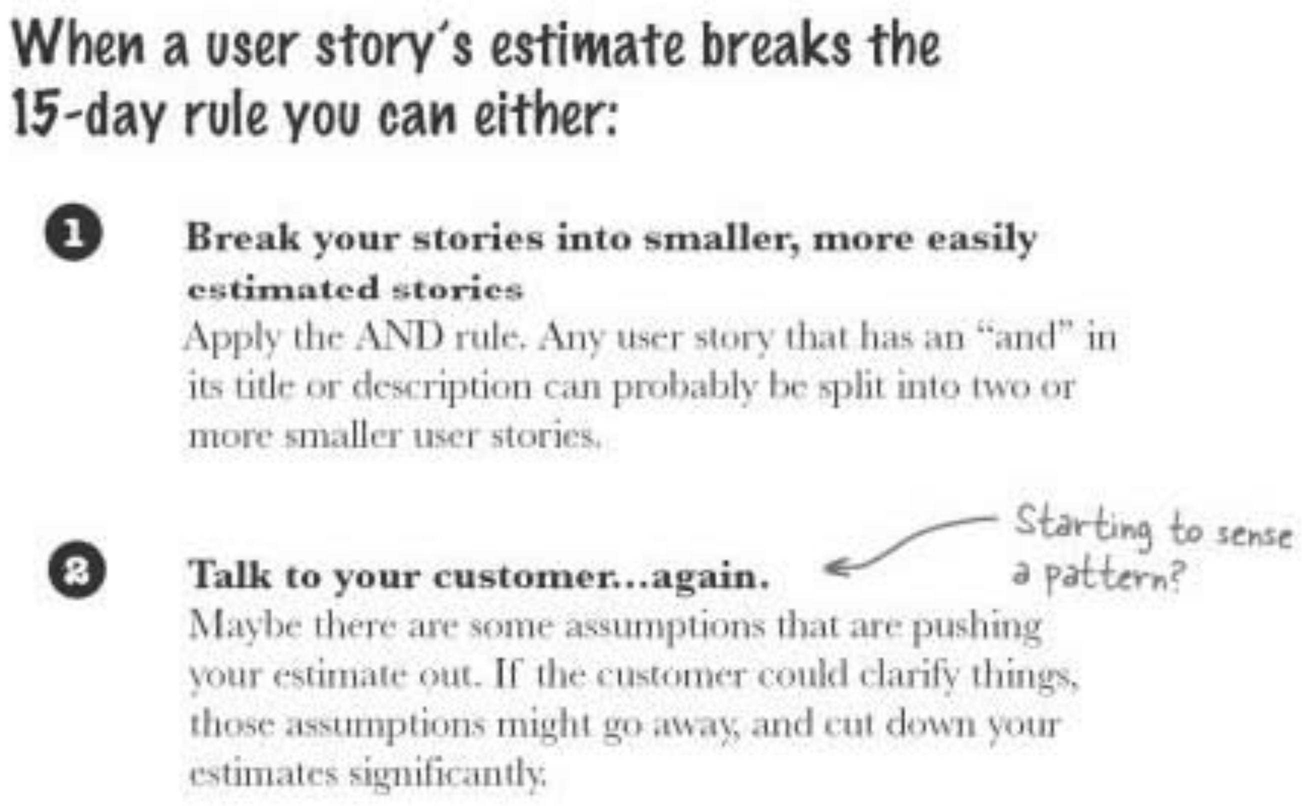
You should be able to check each box for each of your user stories.

Think "by the customer, for the customer"

This means the customer drives each one, no matter who scribbles on a notecard.

If a user story is long, you should try and break it up into multiple smaller user stories (see page 54 for tips).

Requirements Must be Customer-Oriented

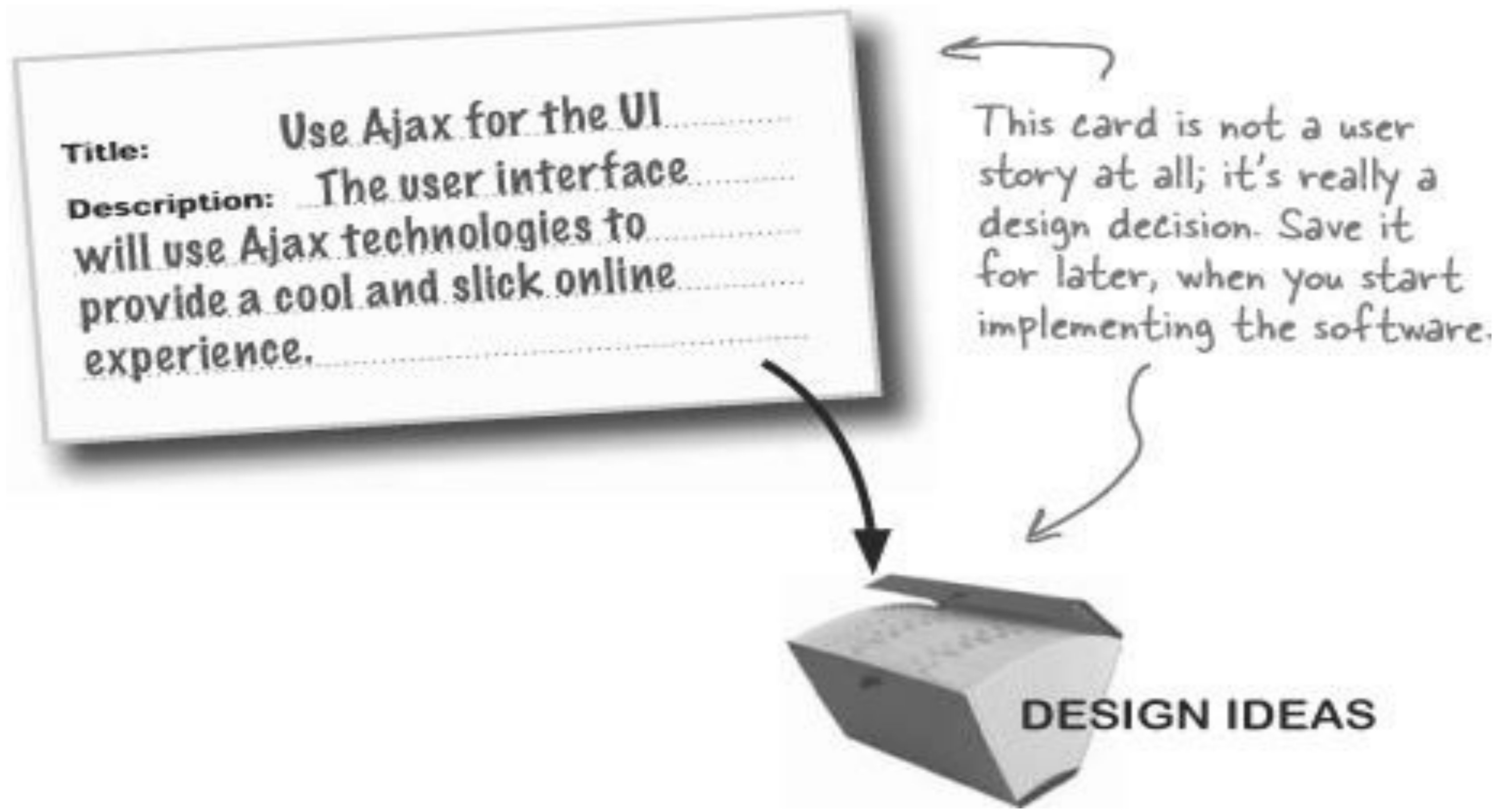


When a user story's estimate breaks the 15-day rule you can either:

1. **Break your stories into smaller, more easily estimated stories**
   Apply the AND rule. Any user story that has an "and" in its title or description can probably be split into two or more smaller user stories.

2. **Talk to your customer...again.** ← Starting to sense a pattern?
   Maybe there are some assumptions that are pushing your estimate out. If the customer could clarify things, those assumptions might go away, and cut down your estimates significantly.

CUSTOMER ORIENTED?


Title: Use Ajax for the UI

Description: The user interface will use Ajax technologies to provide a cool and slick online experience.

This card is not a user story at all; it's really a design decision. Save it for later, when you start implementing the software.
DESIGN IDEAS

Requirements Elicitation

- Requirements Sources
  - Goals
  - Domain Knowledge
  - Stakeholder
  - Operational Environment
  - Organizational Environment

" We need to target *customer needs.* "

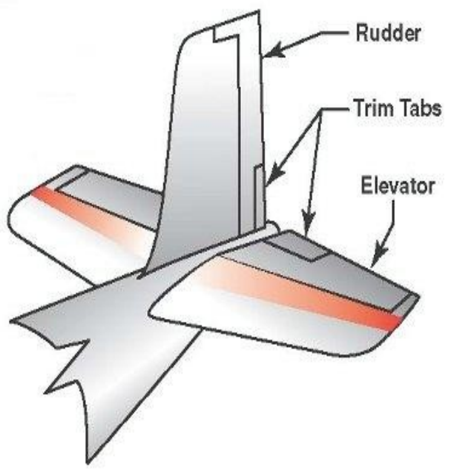
Requirements Elicitation: Goals

- Goals
  - Sometimes called the "business concern" "business case" or "critical success factor"
  - At a high-level, what is the system supposed to do

Requirements Elicitation: Domain Knowledge

- Domain Knowledge
  - Software engineer must have, or have access to, domain-specific knowledge
  - Stakeholder may not have all the information (they just know what they need it to do, not how to do it)
  - Example:
    - Pilot wants to "turn left"
    - Software must: calculate the

rudder position



Requirements Elicitation: Stakeholders

- Stakeholders
  - The software engineer must consider the viewpoints of all stakeholders

Requirements Elicitation: Operational Environment

- Operational Environment
  - Software engineer must gather requirements related to the operational context
- Example:
  - Timing of the landing gear
  - Just getting the job done doesn't suffice,

  you have to do it at the correct time!

Requirements Elicitation: Organizational Environment

- Organizational Environment
  - Software engineer needs gather requirements related to the business/organizational context
- Example:
  - Building a web application optimized for Chrome
  - The organization has standardized on IE

REQUIREMENTS: ELICITATION TECHNIQUES

- Interviews
- Scenarios
- Prototypes
- Facilitated Meetings
- Observation

Elicitation Techniques: Interviews

- Ask the stakeholders what they want
  - Must ask good questions
  - Must have engaged stakeholders

Elicitation Techniques: Scenarios

- Sometimes called "user stories" or "use cases"
- Can be diagramed (in coming weeks)

ELICITATION TECHNIQUES: PROTOTYPES

- Similar to scenarios, can help in clarifying unclear requirements
  - Mock-up
  - Beta-test versions of software

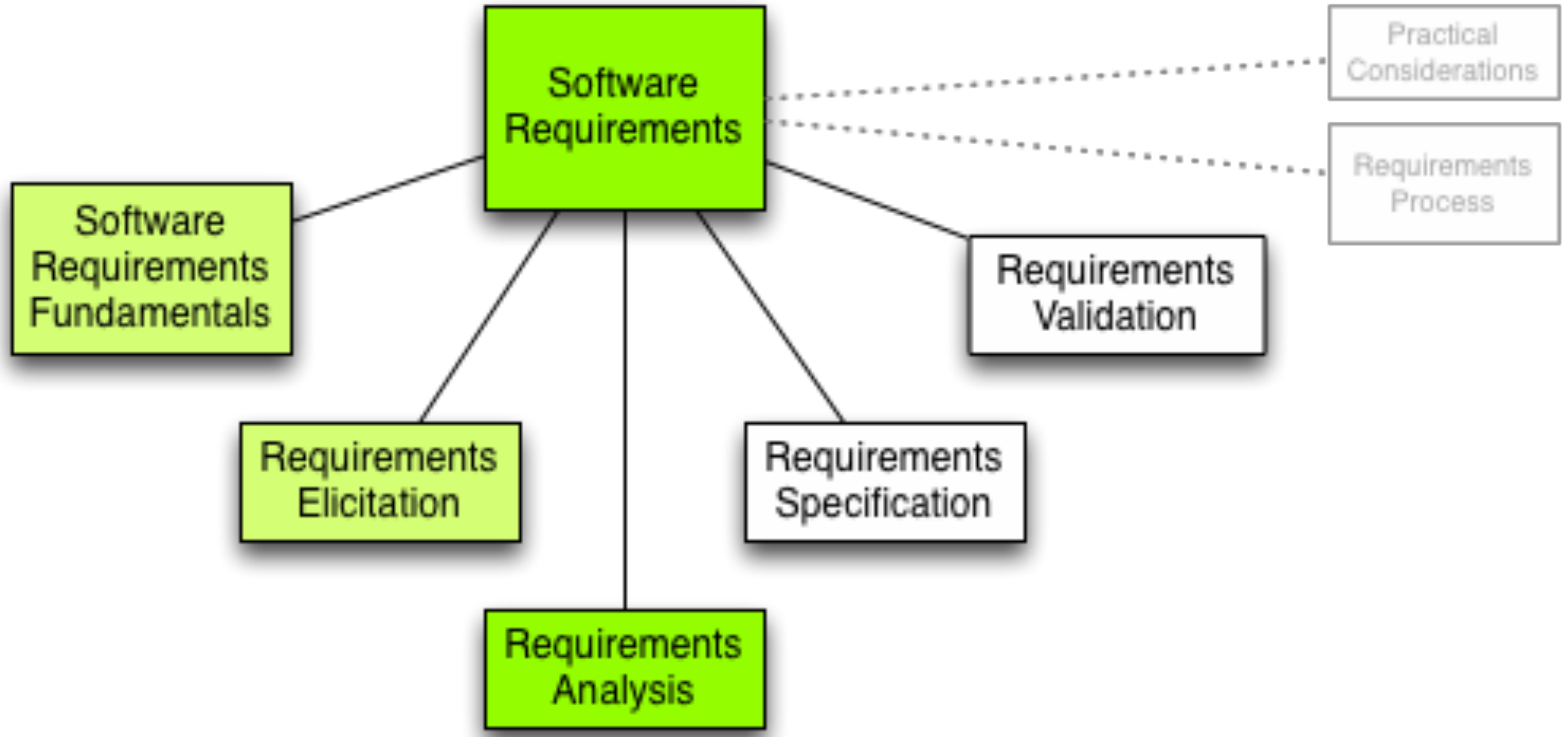
Elicitation Techniques: Meetings

- Sometimes called "blueskying" or "brainstorming”
- Sometimes exposes conflicts (that the facilitator can arbitrate)

Elicitation Techniques: Observation

- Software engineer is involved in the use processes

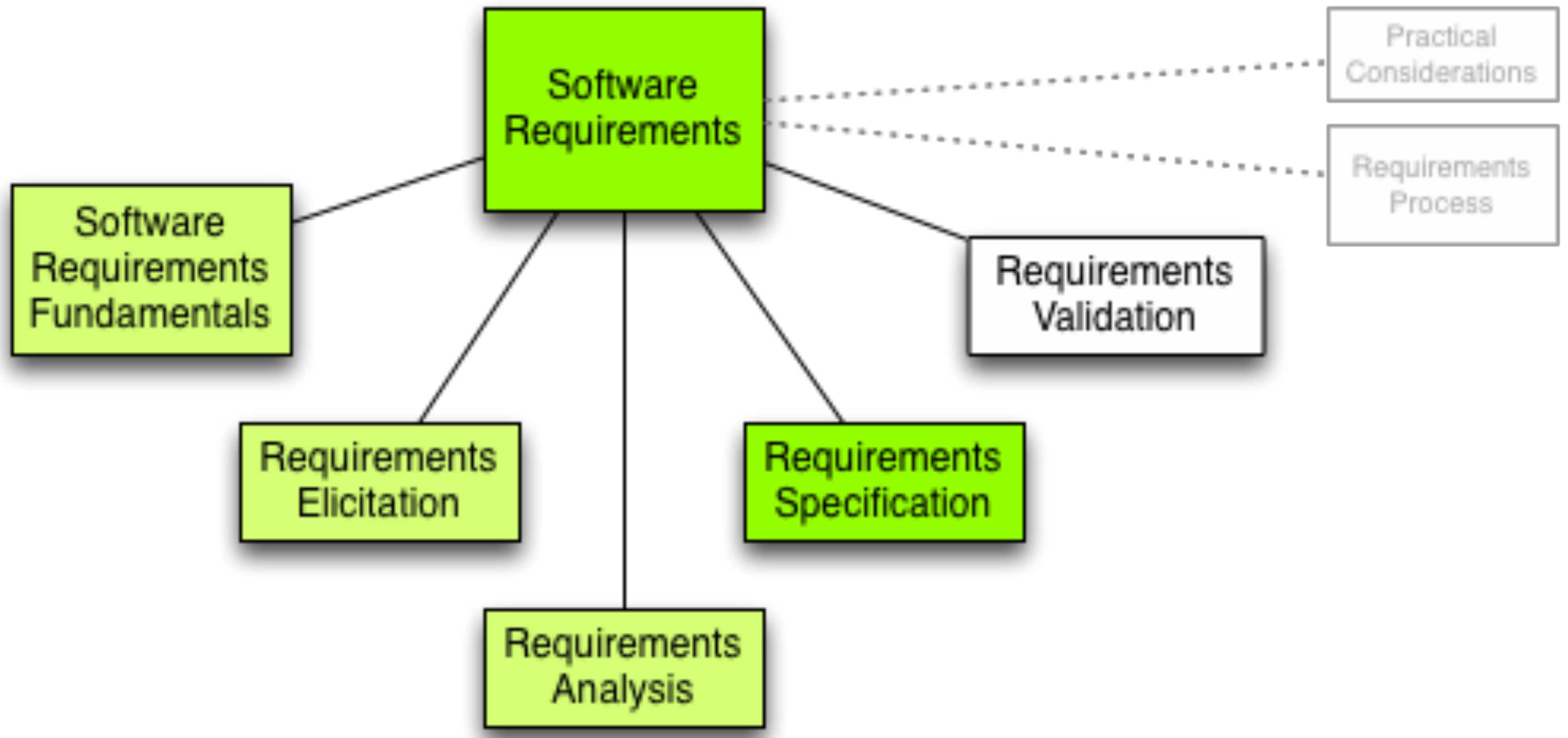
SWEBOK


Software Requirements
Practical Considerations
Requirements Process
Software Requirements Fundamentals
Requirements Validation
Requirements Elicitation
Requirements Specification
Requirements Analysis

Requirements Analysis

- After some requirements are gathered you must analyze
  - Resolve conflicts between requirements
  - Elaborate on high-level requirements to derive software requirements

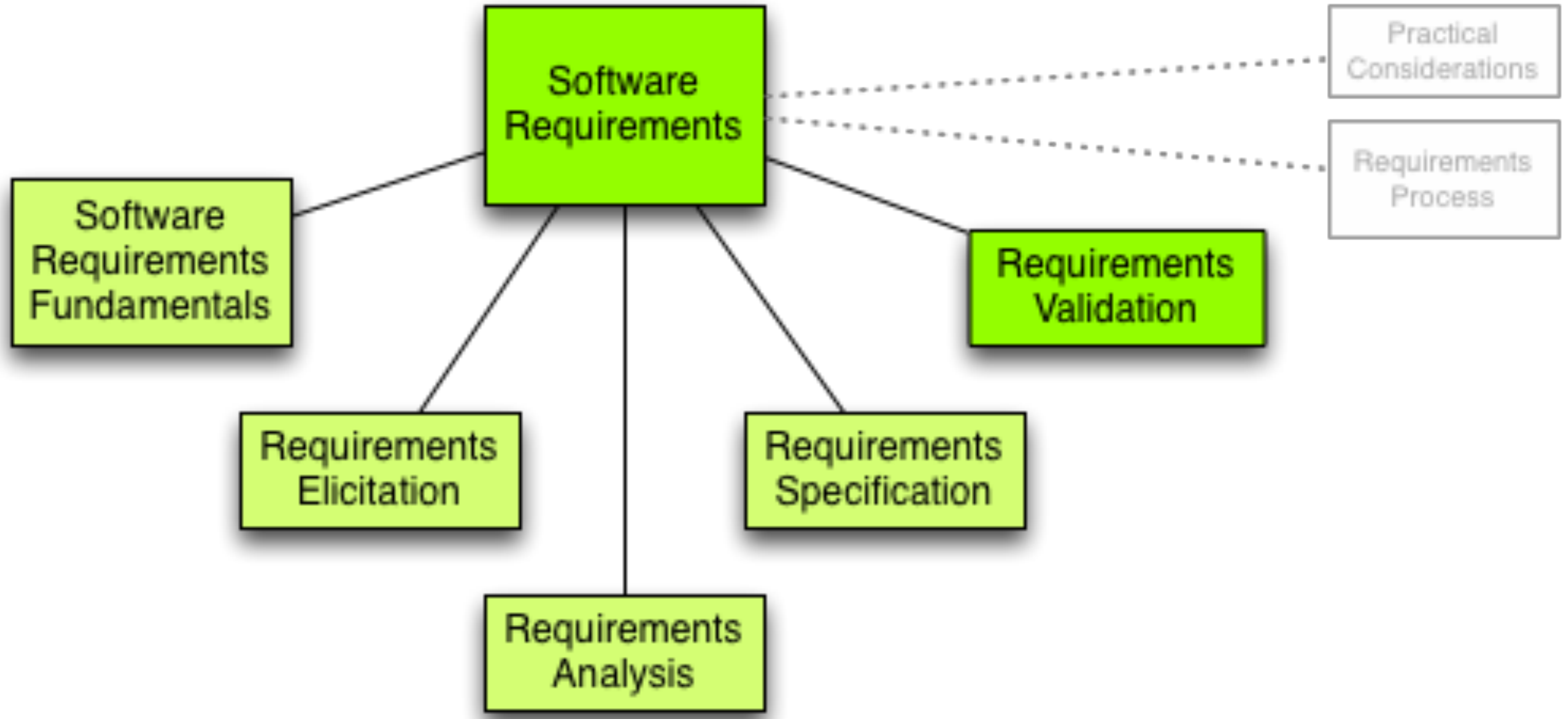
SWEBOK


Software Requirements
Software Requirements Fundamentals
Requirements Elicitation
Requirements Analysis
Requirements Specification
Requirements Validation
Practical Considerations
Requirements Process

REQUIREMENTS SPECIFICATION

- "Software requirements specification" typically refers to the production of a document, or its electronic equivalent, which can be systematically reviewed, evaluated, and approved. [SWEBOK]
- Agreement between the customer and the software engineer
- Specification can be formal or informal
  - For safety-critical systems, a precise formal language may be appropriate
  - Often a specification written in natural language is sufficient

SWEBOK


Software Requirements
Practical Considerations
Requirements Process
Software Requirements Fundamentals
Requirements Validation
Requirements Elicitation
Requirements Specification
Requirements Analysis

REQUIREMENTS VALIDATION

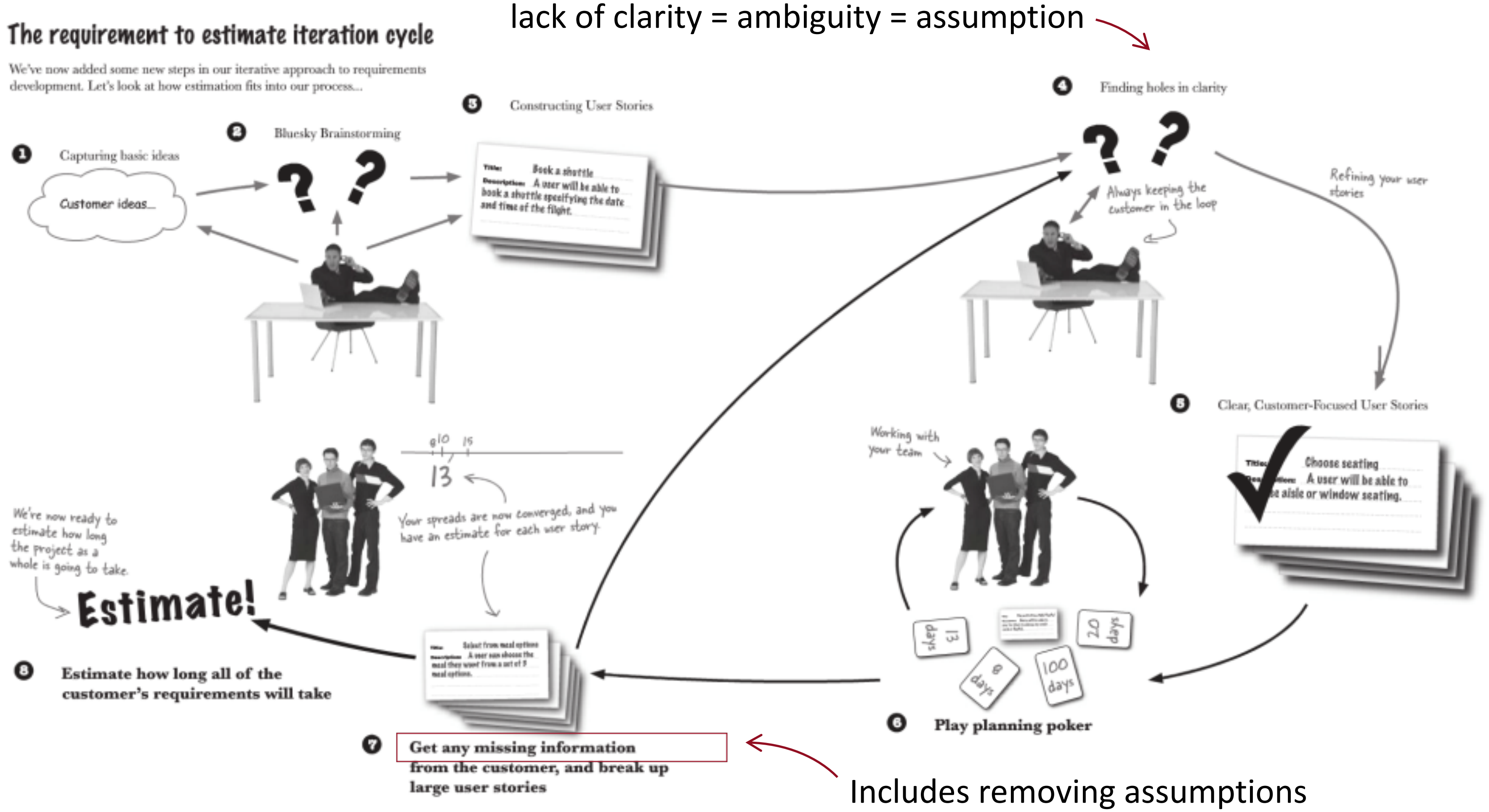
- Validation and Verification
  - Validation: Are we building the right product?
  - Verification: Are we building the product right?

Requirements Validation

1. Requirements Reviews
2. Prototyping
   - "Is this what you meant?"
3. Model Validation
   - If you develop a formal model, prove with formal logic
4. Acceptance Tests
   - How do we verify each (functional) requirement

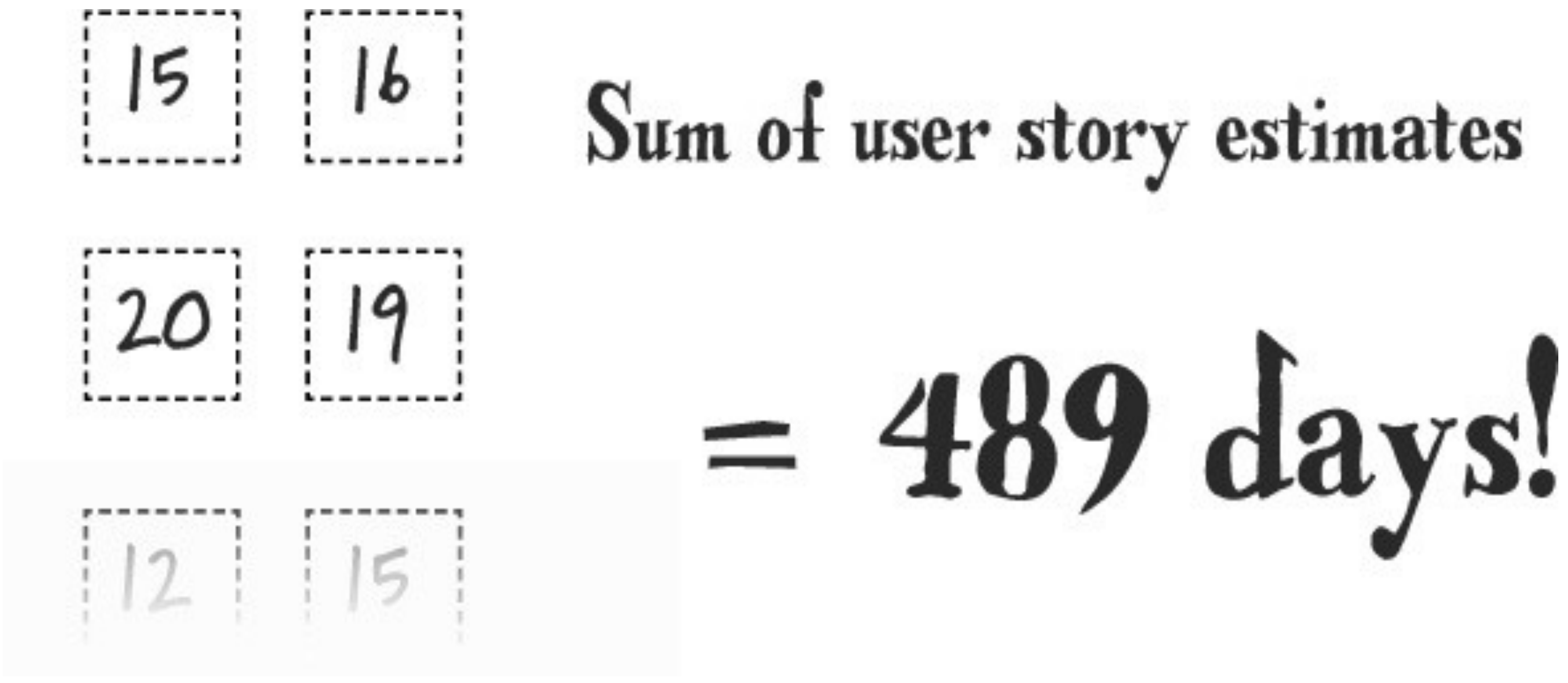
The Requirements Cycle (Process, Algorithm)

lack of clarity = ambiguity = assumption


The requirement to estimate iteration cycle
We've now added some new steps in our iterative approach to requirements development. Let's look at how estimation fits into our process...
1 Capturing basic ideas
Customer ideas...
2 Bluesky Brainstorming
3 Constructing User Stories
Title: Book a shuttle
Description: A user will be able to book a shuttle specifying the date and time of the flight.
4 Finding holes in clarity
Always keeping the customer in the loop
Refining your user stories
5 Clear, Customer-Focused User Stories
Title: Choose seating
Description: A user will be able to choose aisle or window seating.
Working with your team
13 days
20 days
8 days
100 days
6 Play planning poker
7 Get any missing information from the customer, and break up large user stories
Title: Select from meal options
Description: A user can choose the meal they want from a set of 3 meal options.
Your spreads are now converged, and you have an estimate for each user story.
13
We're now ready to estimate how long the project as a whole is going to take.
Estimate!
8 Estimate how long all of the customer's requirements will take

Includes removing assumptions

TOTAL PROJECT ESTIMATE



15 16

20 19

12 15

Sum of user story estimates

= 489 days!

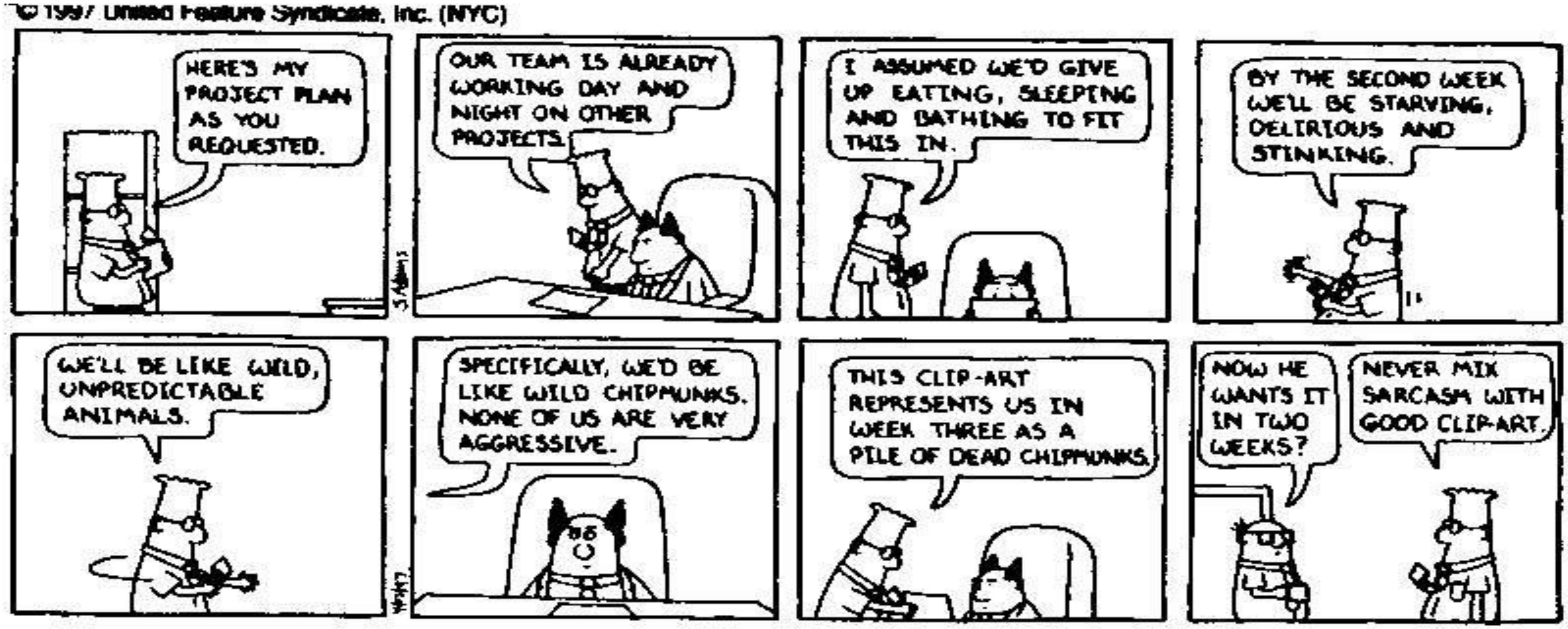
WHAT IF TOO LONG?

No kidding! That's way too long, By the time you've developed the software, my competition will have beaten us into the ground!

What If Too Long?

- Back to drawing board?
- Just ask the customer how long they think?
- Answer:
  - Project Planning! (Next Topic)

NEXT CLASS – PROJECT PLANNING


© 1997 United Feature Syndicate, Inc. (NYC)
HERE'S MY PROJECT PLAN AS YOU REQUESTED.
OUR TEAM IS ALREADY WORKING DAY AND NIGHT ON OTHER PROJECTS.
I ASSUMED WE'D GIVE UP EATING, SLEEPING AND BATHING TO FIT THIS IN.
BY THE SECOND WEEK WE'LL BE STARVING, DELIRIOUS AND STINKING.
WE'LL BE LIKE WILD, UNPREDICTABLE ANIMALS.
SPECIFICALLY, WE'D BE LIKE WILD CHIPMUNKS. NONE OF US ARE VERY AGGRESSIVE.
THIS CLIP-ART REPRESENTS US IN WEEK THREE AS A PILE OF DEAD CHIPMUNKS.
NOW HE WANTS IT IN TWO WEEKS?
NEVER MIX SARCASM WITH GOOD CLIP-ART.