Design Activity

(yes you get credit for this)

Stage 1: Architecture

- Come up with an architecture for your sample system
- What are the major (like, HUGE) components?
- How do they fit together?

What we (basically) did:
we didn't think too hard about this, we just tried to identify large elements in the system (client; server; layers; etc). and then tried to figure out how they might link together.

Stage 2: Detailed Design

- What are the smaller elements (classes) in the system?
- What do those elements basically do?
- How are they hierarchically organised (inheritance)
- How are they associated with one another?
- How do they fit into the architecture (maybe each architectural component is a package?)

What we (basically) did:

we applied a naive heuristic for achieving a design. We looked at the user stories and acceptance criteria and…

1. found classes by looking for nouns
2. found methods by looking for verbs
3. found fields by looking at attributes of nouns
4. derived associations and specialisation relationships between classes
5. didn't include any elements not explicitly needed for the user story and acceptance criteria

Stage 3: Playing out a user story

- Starting with the “trigger” of a user story, draw the interaction diagram for how the classes active in the story communicate

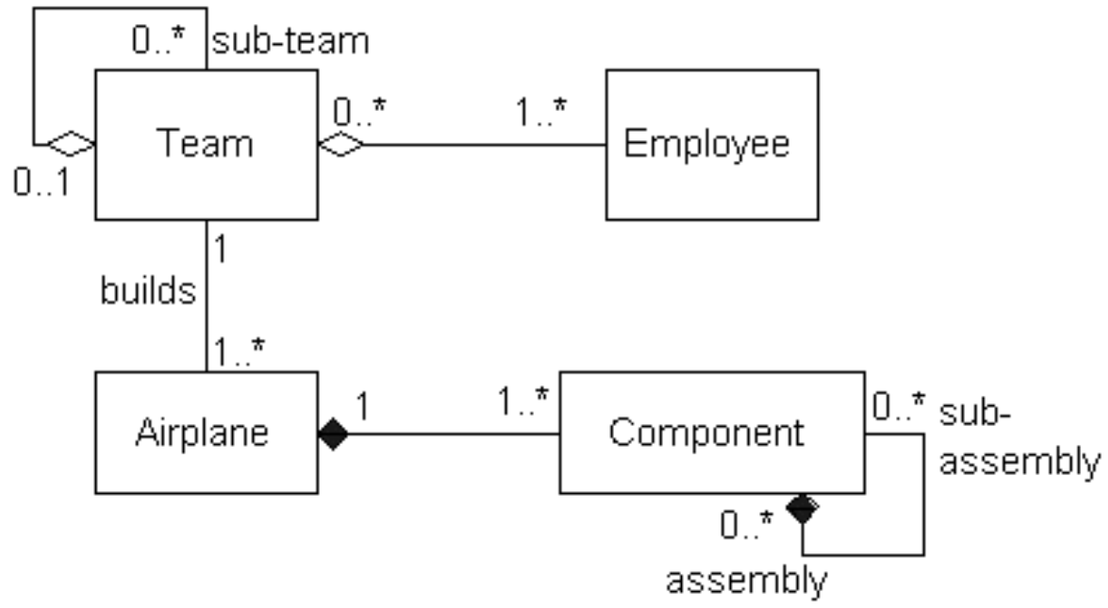
Now pass everything down to me!

- Let’s take a look!

What can Class-Relationship Diagrams Tell Us:

Facts this diagram reveals:

- An airplane is made up of components
- An airplane is a composition of 1 to many components
- Each airplane is required to have one or more components
- A component is associated with a single airplane
- A component can be composed of zero to many other components
- A component can be sub-assembled into 0 to ma.., other components
- Once an airplane is destroyed all the components also need to be destroyed
- A team is made up of employees and sub-teams
- A team consists of any number of subteams; a subteam may or may not have a parent
- A team builds at least one airplane
- Each airplane is built by exactly one team
- A team can be an aggregation of 0 to many subteams
- A subteam can be part of 0 or 1 team
- A team can have 1 or more employees
- An employee can be a member of 0 to many teams
- When a team is destroyed the employee can still exist (the instance doesn't have to be destroyed)



What the diagram doesn't tell us:

- Why are teams and sub-teams in weak aggregation when the similar assembly and subassembly are in strong aggregation?
- What is this system supposed to do?
- What is the difference between the shaded and non-shaded diamond?
- Does the diagram reflect that an Employee can't work on an Airplane unless they belong on a team?
- What do the reflexive relationships mean?