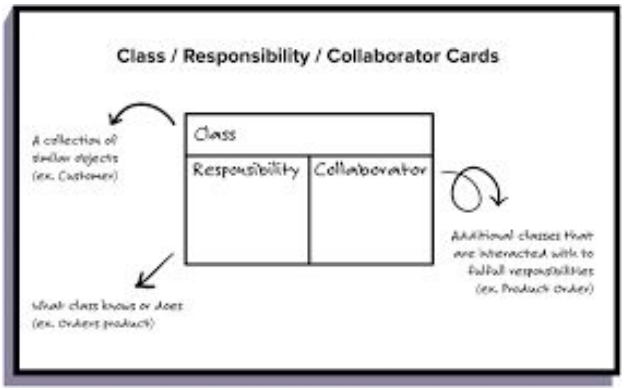
Lecture Notes
December 10, 2020

Review of CRC. (You need to use CRC for the second assessment. Today's demos will be after the CRC review.)

CRC = Class Responsibilities Collaborators

Like user stories, CRC can be written on 3x5 (or 4x6) index cards, so often known as "CRC cards"

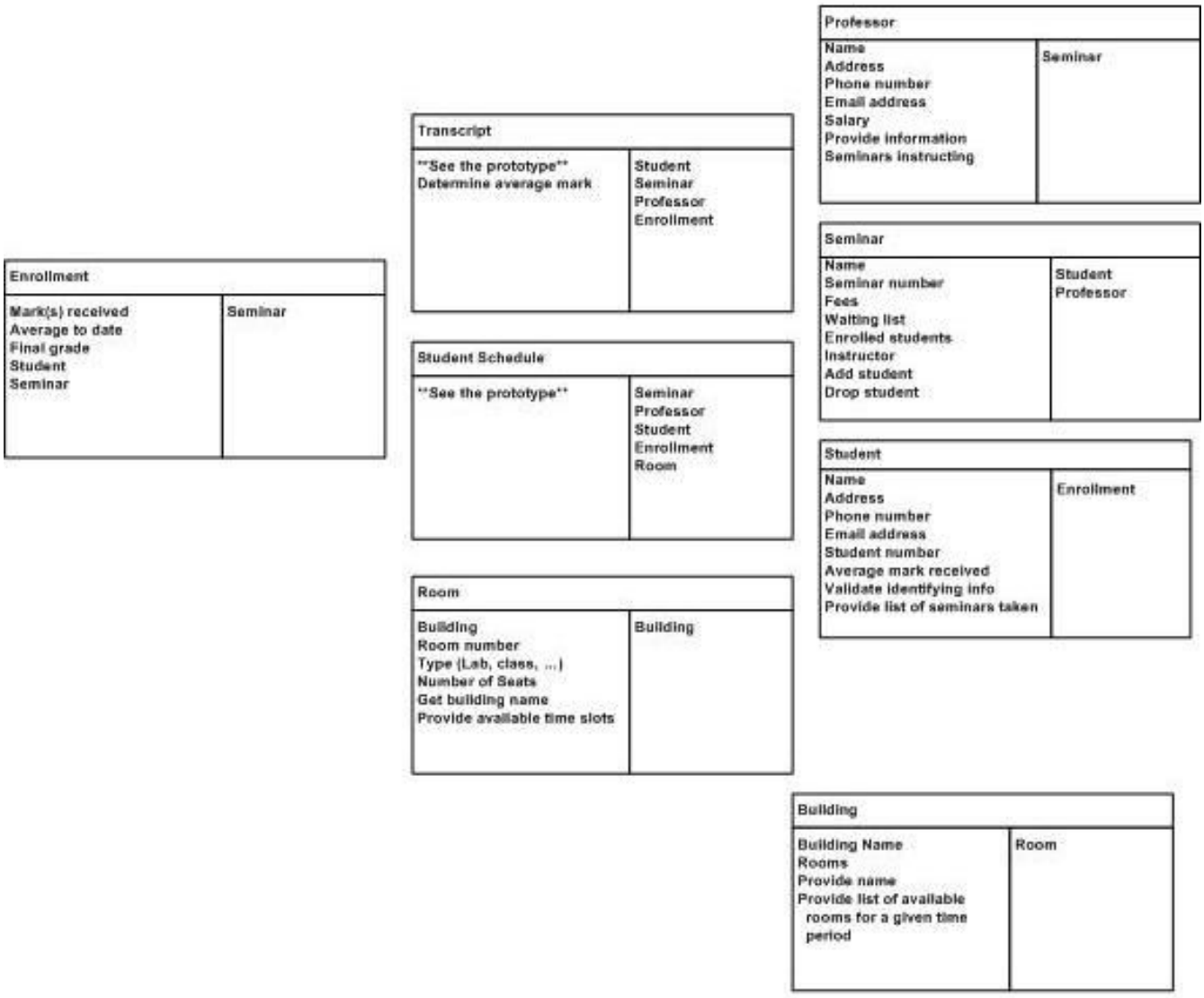


Each *class* is a thing, entity or object

Each *responsibility* is some action the entity needs to do

Collaborators are other classes the entity interacts with, communicates with, contains, knows about, or that otherwise help it perform one or more responsibilities

CRC focuses on the *purpose* of each entity rather than its processes, data flows and data stores (procedural design)



How do we know what the classes (entities) are?

Start with user stories or use cases

Find the *nouns* other than the user (user role) - classes should be the entities that “do” actions, not the actors who initiate actions

When “the system” does something, that means some class in the system does it - there should not be any global master that does everything

Beware hidden nouns:
“the thing is moved” = “SOMETHING moves the thing”. SOMETHING is either an actor or another class. If another class, the thing is SOMETHING’s collaborator as well as itself a class

Beware nouns as verbs:
“the thing does SOMETHING”. Do-SOMETHING is (probably) the thing’s responsibility

Some nouns may be attributes of entities, not themselves entities, e.g., “the length of a rectangle” - here the rectangle is an entity and the length is an attribute of the rectangle, length has no meaning as a standalone entity

How do we know what the responsibilities (actions) are?

Find the *verbs*

Focus on WHAT gets done, not HOW it gets done

Turn passive verbs into active verbs, as above
"the thing is moved" = "SOMETHING moves the thing"

Do/does/did and so on is not (usually) a responsibility by itself, the responsibility is to do-SOMETHING

Where do the collaborators come from?

Find any relationships or associations among nouns

Collaborations are not necessarily symmetric, might be one-way. For example, SOMETHING needs to know about the thing in order to move it, but the thing may not need to know about SOMETHING (uses vs. is-used-by)

How does CRC design work?

Develop a set of CRC cards for the current set of user stories (current iteration or upcoming release). Classes and Collaborators do not need to correspond to code classes, types or data structures - can be modules, packages, libraries, even files, databases, devices.

CRC does not need an object-oriented programming language, C is fine

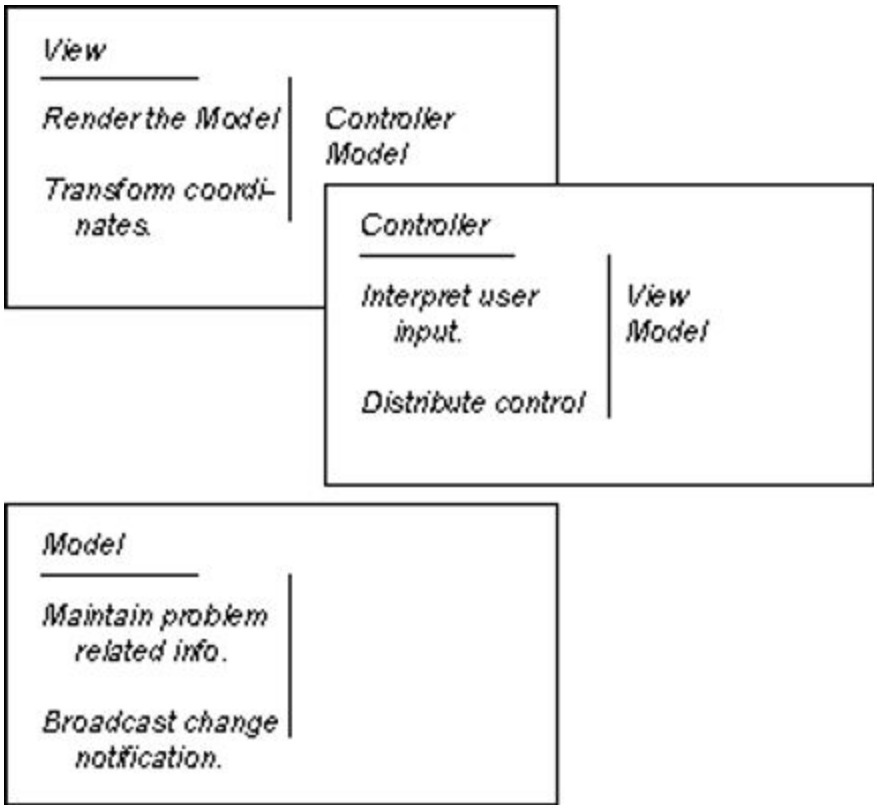
Walk through the user stories conditions of satisfaction or the flows of each use case and *animate* the CRC cards: move index cards around on a table, sticky notes on a board, or use an online program like CRC maker. Make sure everything that needs to be done is a responsibility of some class, and figure out what other classes each class needs to fulfill its responsibilities

Then debug the CRC cards:

- Remember classes are nouns that “do” operations, not the actors who initiate operations
- Split classes with too many responsibilities
- Combine classes with too few responsibilities (but only if they make sense together)
- Remove redundancy

Also remove classes with only CRUD operations (create, read, update, delete) that don't "do" anything, In most cases CRUD operations should be part of some functionality. An exception would be a program whose whole purpose is to interface to a database

UI elements, and views in general, may not fit nicely into CRC. In MVC designs, CRC is concerned with functionality of the business logic - which ought to reside in the model (although there are some variants of MVC where the model is just data, and the business logic resides in the controller)



Volunteer final demos: Alchemist, Falcon, Advance Engineers, YesOK, TaiOne. Please keep your demos *short* to allow enough time for all the teams that want to demo during class

Important resources:

Anything written by Martin Fowler

Anything written in 2018 or earlier by Joel Spolsky, most of his 2019/2020 writing is about his companies and his retirement

Someone mentioned Uncle Bob (cleancoder.com) but the website is largely focused on selling his books and classes and much of his free material is specific to Clojure functional programming - great if you are using Clojure

geeksforgeeks.org

stackoverflow.com for specific questions/topics

softwaretestingfundamentals.com

medium.com has a lot of relevant articles, but is not focused on software engineering

Assignment T6: Final Demo: https://courseworks2.columbia.edu/courses/104335/assignments/491047 due December 10 (tonight)

Second Assessment available 12:01am December 11 (also tonight), due 11:59pm December 14. The deadline for students with an “extended time” arrangement is 11:59pm December 16.

Extra Credit: Optional Demo Video https://courseworks2.columbia.edu/courses/104335/assignments/543277 due December 20