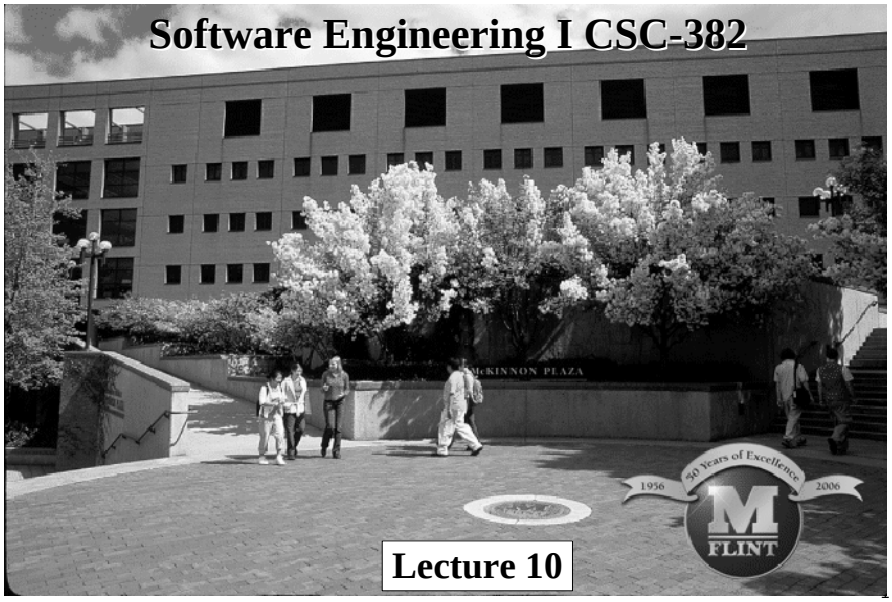
Software Engineering I CSC-382



Lecture 10

Software Engineering I CS-382

- Lecture 10
- What we will cover: (Details of Analysis Modeling)
  - Chapter 8 Sections 8.4 and 8.7 in Pressman
  - Goal is to understand the methods and tools available from Object Oriented Analysis for analysis modeling

Introducing the CRC Card

- CRC stands for: Class, Responsibility and Collaborators
  - **Class**: An object is a person, place, thing, event, or concept
  - **Responsibility**: Anything that a class knows or does
  - **Collaborator**: A class that another class needs to accomplish its purpose
    - In general, a collaboration implies either a request for information or a request for some action.

Introducing the CRC Card II

- The CRC card is a flexible and **informal** tool that can help the above process
  - The text tends to use them later but I find they help bridge the magic gap from Use Cases to candidate classes
  - The resulting set of cards are your Class Model

Elements of CRC Modeling

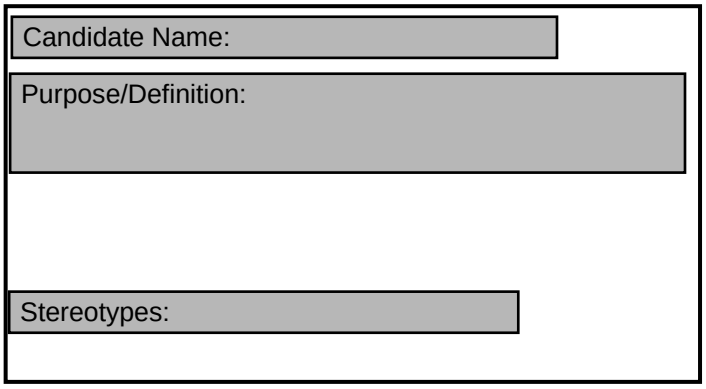
- While it seems a little silly, the use of actual cards helps to emphasize the essence of object-oriented modeling
- Identifies the definition of the responsibilities of the objects
- Clearly shows that everything that is happening in the system is happening as a result of the actions of an object in the

Elements of CRC Modeling II

- Note the focus of CRC modeling is on the ***behavior*** of the objects and not the specific structure of the objects
  - Since we are more interested in behavior than structure we will leave the definition of the attributes of the objects for later and focus on their functionality.
  - The attributes will be clearly defined and reflected in the "***know responsibilities***" of the objects to ***know*** certain things as defined in the CRC cards.

Book's Structure of CRC Card

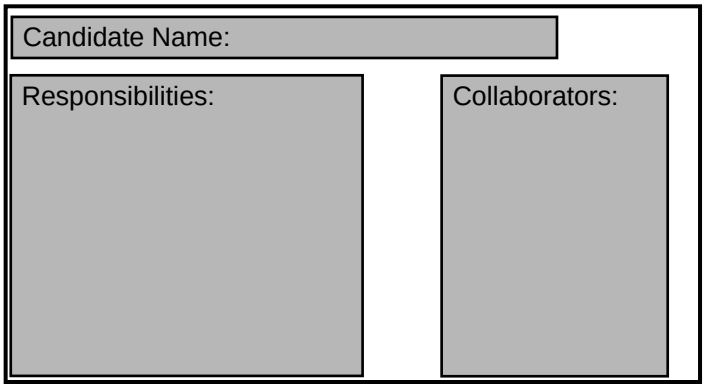
Front of Card:



Can do all on 1 side but this way allows us to focus on the name and purpose and then address the harder issues of responsibilities and collaborators.

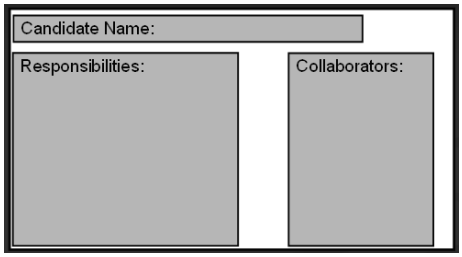
Book's Structure of CRC Card II

Back of Card:

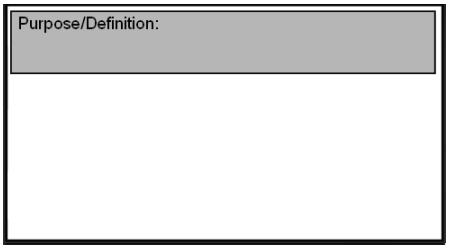


More Common Structure of CRC Card

Front of Card:



Back of Card:



This is often (and originally) used as the CRC
I prefer it since it is simpler and focuses on the main info

Responsibilities

- System intelligence should be distributed across classes to best address the needs of the problem (avoid central controller class)
- Each responsibility should be stated as generally as possible
- Information and the behavior related to it should reside within the same class (good encapsulation)
- Information about one thing should be localized within a single class, not distributed across multiple classes.
- Responsibilities should be shared among related classes, when appropriate.

Collaborations

- Classes fulfill their responsibilities in one of two ways:
  1. A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or
  2. A class can collaborate with other classes.
- Collaborations identify relationships between classes
- Collaborations are identified by determining whether a class can fulfill each responsibility itself

Collaborations II

- Three different generic relationships between classes [WIR90]:
  - the ***is-part-of*** relationship
  - the ***has-knowledge-of*** relationship
  - the ***depends-upon*** relationship

How Do We Define the RC Part of the CRC Now ?

- The specific sequence of ***responsibilities*** of the objects and their required ***collaborations*** can only be seen thru how the system accomplishes its goals
- Recall Use Cases were used to identify key actors, their goals, and the steps they and the system followed to meet these goals
  - Therefore the Use Cases we developed will be an excellent source of information for defining the ***Responsibilities*** and ***Collaborators***

Recipe for Generating CRC Cards

1. Gather up any other requirements information we may know
   - E.g. Product Vision Statements, Use Cases, etc.
2. Brainstorm on all the possible classes in the system
   - Use the ideas defined earlier for possible sources of class/objects
3. Filter this list
   - Combine related classes, delete redundant ones
   - Develop a good 1 sentence description of each class
   - Now fill in the ***Name*** field and the ***Description*** fields on the CRCS

Recipe for Generating CRC Cards II

4. Walk thru the scenarios
   - Scenarios define the functions of the system
   - Recall the Use Cases also defined functions that the actors on the Use Case perform
     - Organize your Use Cases into thematic groupings
     - Start with the simplest Use Cases and work to the more complex
   - As each action on the Use Case is performed, it corresponds to a responsibility and is assigned to a particular CRC
     - Write the responsibility down on the card

Recipe for Generating CRC Cards II-II

4. Walk thru the scenarios (continued)
   - At each action also identify any collaborators the class needs to accomplish the task and write them on the CRC
   - Don't forget to have that collaborating CRC write its responsibility down when it is identified at the same time

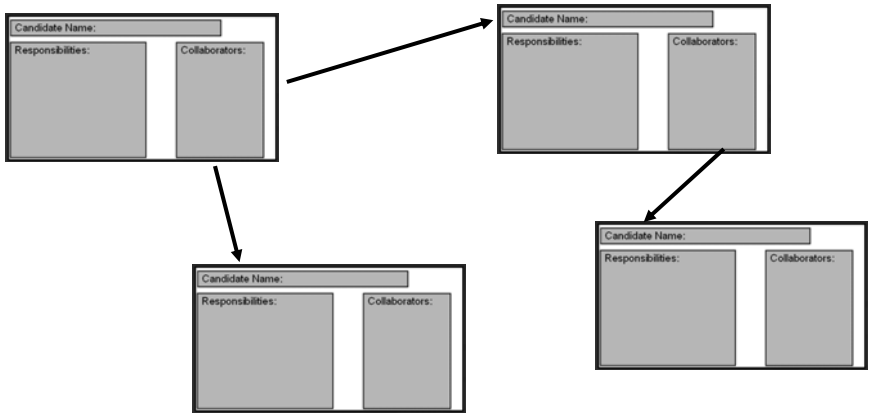
Recipe for Generating CRC Cards III

5. Group the Cards
   - Once the scenario modeling is complete group the cards on a table
   - The proximity is defined by how much they collaborate
     - Two highly collaborating cards are placed close together
     - Also possible hierarchies (sub-classes) are grouped together
   - This grouping by collaboration can help to begin to identify subsystems within the primary system.

Recipe for Generating CRC Cards III-II

5. Group the Cards (continued)
   - One possible extra step is to develop a ***Collaboration Drawing***
     - Tape all the CRC cards to a white/black board
     - Draw lines of collaboration between the cards
     - Hopefully this drawing will also show some clustering of the paths of collaboration within the system

Collaboration Drawing



Some Comments on the CRC Process

- This is a very dynamic session.
  - The designers are each assigned some set of CRCs and they 'become' the object in the scenario
- To keep it moving and animated:
  - Handle the normal scenarios first
  - Once a model to handle these is done then go back and see how it fits (and modify it) for the exception cases in the Use Cases
  - If there are two ways to do something and you cant decide then just choose one and try it (it is easy enough to redo the cards another way)

Some Comments on the CRC Process II

- To keep from getting lost always have the person holding the card that is in action to hold it in the air
  - Provides good visual cue as to where the action is
- Feel free to add a class whenever it appears one is missing to accomplish some needed collaboration

Recap of the Recipe for Generating CRC Cards

1. Gather up any other requirements information we may know
2. Brainstorm on all the possible classes in the system
3. Filter this list for duplicates and write descriptions
4. Walk thru the Use Case scenarios (normal scenarios first)
5. Group the Cards (maybe use Collaboration Drawing)

In-class CRCs

What do we do first??

Then…??

For Next Class

- Midterm Review
- Midterm on Wednesday!!
  - Bring ½ of a 8 ½ x 11 sheet of paper for notes