Prifysgol Abertawe
Swansea University

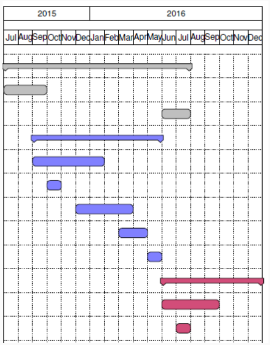
CS-230 Software Engineering

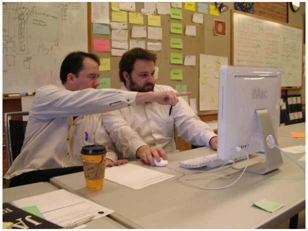
L05: Class Design and Responsibilities

Dr. Liam O'Reilly

Semester 1 – 2020

Previously in CS 230...





Team Work, Gantt Charts, Risk Analysis

Previously in CS 230...

Lets go a bit further back...

```
Book
-----------------------------------------------------------------
  – pages : BookPage []
  – softCover : boolean
  – author : string
  – title : string
-----------------------------------------------------------------
+ Book (in author : string, in title : string)
+ toPage (in pageNum : integer) : BookPage
+ skimPages (in start : integer, in end : integer) : integer
– makeNewPage (in contents : string) : BookPage
```

Design... No Lumping Together!!!

Previously in CS 230... (2)

- A class/object abstracts away...
  - The inner workings (data & operations),
  - into a single item to be used as a part of a system.
- A class is...
  - A description of values that can be stored.
  - A description of the operations on those values.
- An instance of a class is...
  - An instance (in memory) of those values that can be modified.
  - Operations work on an instance (i.e., are executed on those values).

Previously in CS 230... (3)

- What are each of the following
  - Public...
    - Accessible outside this class.
  - Private...
    - Accessible only inside this class.
  - Static ...
    - A single copy outside all instances/associated with the class.
  - Constructors...
    - Special “methods” to construct new instances.

Previously in CS 230... (4)

- We understand what classes are.
- We know how to specify them in UML.
- But, how do we know when to create a class?
- The first step in design!

Responsibilities

Responsibilities

- Two kinds of responsibilities
  - knowledge maintained by object *attributes*.
  - actions a class can perform *behaviours/operations*.
- Should represent purpose of the class in system.
- Define services provided by the class to system.
- Two rough ideas for types of classes:
  - Information objects can store information and be returned to client code.
  - Action objects can perform operations for client code.
- Only public services count.
- private knowledge may need to be implemented by a class, but definition delayed.
- Concentrate on what the class does not how it's done.
- I.e., How it interacts with other classes.

Identifying Classes and Responsibilities

- Examine nouns in requirements specification:
    - May become a class.
    - May become an attribute of a class, or a part of a composite class.
    - Information words can imply "how much, how many, how big", etc.
- Examine verbs in requirements specifications:
    - Active voice verb may indicate a responsibility.
    - Turn passive voice sentences into active voice and examine.
        - `https://writing.wisc.edu/Handbook/CCS_activevoice.html`
- Perform a system walk-through (Use Case Diagrams in UML terms):
    - Make sure that all system responsibilities identified.

Assign Responsibilities

- Identified responsibilities must be assigned to classes.
- Examine context in which responsibility was identified.
  - Use requirements specification, as well as class definitions.
- Some assignments will be obvious, others will require some thought.
  - Distribute system intelligence evenly.
  - State responsibilities as generally as possible.
  - Keep behaviour (actions) with related information.
  - Keep information about one thing in one place.
  - Responsibilities can be shared among related objects.
- Each class should have one main purpose, one idea, one main responsibility.

System/Class Intelligence

- System Intelligence:
  - What the system knows – information it stores.
  - What actions the system can perform – its functions.
  - Relation to other systems.
- Class intelligence:
  - What does the class know – information it stores.
  - What services does it provide (server view).
  - What services does it use (client view).

Responsibility Guidelines

- State responsibilities as generally as possible.
    - Instead of stating that a LineElement knows how to draw a line and RectangleElement knows how to draw a rectangle.
    - We say that both know how to draw themselves.
- May lead to more general classes (i.e., superclasses)
- If an object maintains particular information, it should be responsible for operations on that data and vice versa.

Responsibility Guidelines (2)

- Keep information about one thing in one place.
    - Maintenance of specific information should not be shared.
    - Leads to duplication of information, which may lead to inconsistency.
- If more than one object must know the information.
    1. Assign the information to one object/class if there is one that has few other responsibilities.
    2. If classes requiring the information have few responsibilities, collapsed into a single class?
    3. Create a new class to take the responsibility of managing the information. Other classes can collaborate with this new class to access the information.

Shared Responsibilities

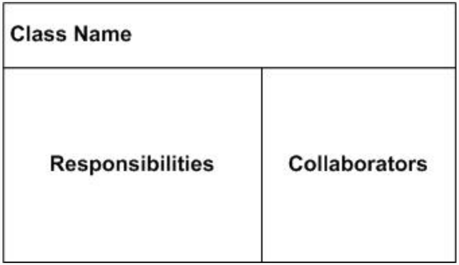
- Some responsibilities need to be shared among several classes. They are compound responsibilities.
- Split into more specific components to distribute intelligence.
- Indicative of relationships between the classes.
    - Composition or Aggregation relationships.
    - Hierarchical relationships.

Unassigned Responsibilities

- Difficulties in assigning responsibilities can occur:
  1. A class is missing:
     - May need to add a class to handle a set of unassigned responsibilities. Have you identified a new sort of entity?
  2. Responsibility could be assigned to more than one class:
     - Sometimes the assignment is not obvious. Make a tentative arbitrary assignment. Try a walk-through. See how it works. Try an alternative. See how that works.

Specifying Responsibilities: CRC Cards

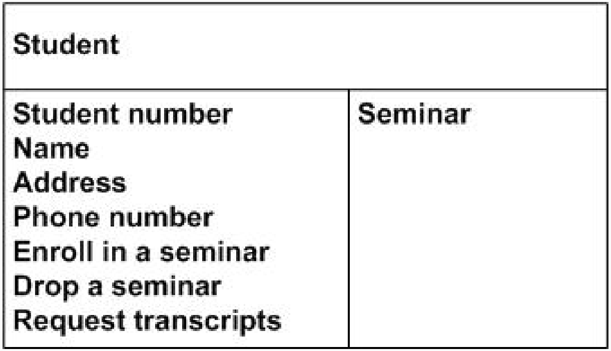
- Class Responsibility Cards (CRC) help with the design of classes and their collaborations.



http://www.agilemodeling.com/artifacts/crcModel.htm

- Responsibilities – attributes and behaviours/operations for this class.
- Collaborations – other classes that it needs to work with.

CRC Cards Example

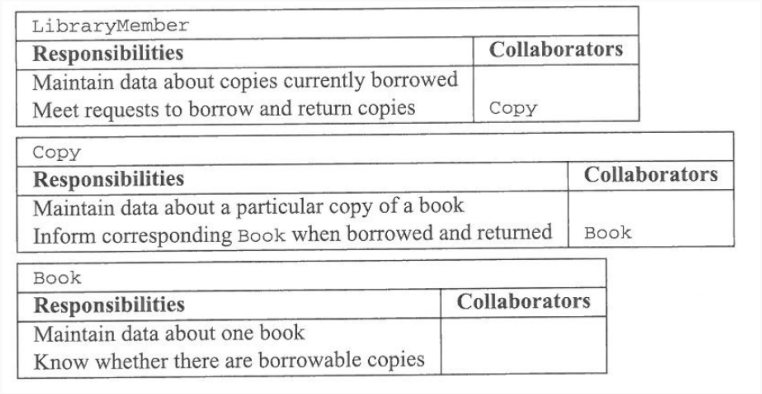


Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

http://www.agilemodeling.com/artifacts/crcModel.htm

(Maybe you decide Address should be its own class and then Student would also collaborate with the Address class.)

CRC Example 2



LibraryMember	
Responsibilities	**Collaborators**
Maintain data about copies currently borrowed	
Meet requests to borrow and return copies	`Copy`

Copy	
Responsibilities	**Collaborators**
Maintain data about a particular copy of a book	
Inform corresponding `Book` when borrowed and returned	`Book`

Book	
Responsibilities	**Collaborators**
Maintain data about one book	
Know whether there are borrowable copies	

Using UML, P. Stevens and R. Pooley

Design Approach

1. Think about what data and operations **belong together**.
    - These are your candidate classes.
2. Draw up CRC cards for your preliminary class design.
3. Think about and reflect on the design:
    - If a class has too many responsibilities, divide it up.
    - If it has too few responsibilities, may be it should be merged with other classes.
4. When settled, begin a UML design to precisely specify responsibilities and collaborations.

Exercise

Designing an object oriented system for the game of Connect 4.

- Write CRC cards specifying the classes in the system.
- Choose one of your classes and draw it in UML.

Exercise 2

- You designing an object oriented system to manage purchases in a store.
    - Customers can come in and place an order of items.
    - Items have a price and the cost of the customer's order can be calculated.
    - Customers have personal information: name, address etc.
    - There is also facilities to check the inventory of items.
- Write CRC cards specifying the classes in the system.
- Choose one of your classes and draw it in UML.