Object Oriented Analysis & Design review concepts

Topic # 13

Homogenization of system

CRC

Homogenization of the system

- The word homogenize means to blend into a smooth mixture, to make homogeneous.
- In parallel design process, several stimuli with the same purpose or meaning are defined by several designers. These stimuli should be consolidated to obtain as few stimuli as possible. It is called homogenization.
- Homogenize - to change (something) so that its parts are the same or similar.

Need of Homogenization

- It is often seen same classes doing more or less the same functionality with different names.
- This is a serious design issue… these names conflicts must be resolved.
- Homogenization is an ongoing process.

Components of Homogenization

- Combining Classes
- Splitting Classes
- Eliminating Classes
- Consistency Checking
- Scenario Walk-Through
- Event Tracing
- Documentation Review

Combining Classes

- If different teams are working on different scenarios, a class may be called by different names. The name conflicts must be resolved.
- This is accomplished mainly through model walk-throughs. i.e.
  - Examine each class along with its definition.
  - Examine the attributes and operations defined for the classes, and look for the use of synonyms.
  - Once you determine that two classes are doing the same thing, choose the class with the name that is closest to the language used by the customers.

Splitting Classes

- Classes should be examined to determine if they are following the golden rule of OO, which states that a class should do one thing and do it really well.
- They should be cohesive;
- Example :
- A StudentInformation class contains:
  - Info about student Actor
  - And info about the courses that student has successfully completed.
- This is better modeled as two classes—StudentInformation and Transcript, with an association between them.

Eliminating Classes

- A class may be eliminated altogether from the model.
- **This happens when:**
  - The class does not have any structure or behavior
  - The class does not participate in any use cases
- **Guideline:** Examine control classes.
- Lack of sequencing responsibility may lead to the deletion of the control class.
- This is especially true if the control class is only a passthrough— that is, the control class receives information from a boundary class and immediately passes it to an entity class without the need for sequencing logic.

Consistency Checking

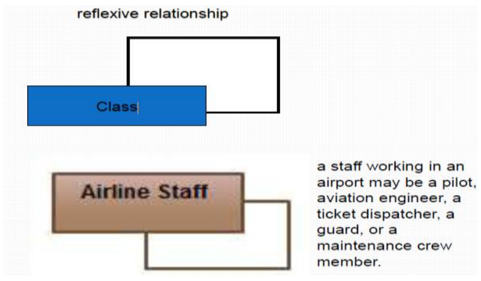
- Consistency checking is needed since the static view of the system, as shown in class diagrams, and the dynamic view of the system, as shown in use case diagrams and interaction diagrams, are under development in parallel.
- Because both views are under development concurrently they must be cross-checked to ensure that different assumptions or decisions are not being made in different views.
  - It should be verified that each view of the system has the same assumptions/ decisions and terminology used.
- Consistency checking is done by forming a team of 5 to 6 members
  - Analyst, designer, customer/customer representative, domain expert

Scenario Walk-Through

- A primary method of consistency checking is to walk through the high-risk scenarios as represented by a sequence or collaboration diagram.
- Since each message represents behavior of the receiving class, verify that each message is captured as an operation on the class diagram.
- Verify that two interacting objects have a pathway for communication via either an association or an aggregation.
- Especially check for reflexive relationships that may be needed since these relationships are easy to miss during analysis. Reflexive relationships are needed when multiple objects of the same class interact during a scenario.
- Finally verify that each object in the sequence/ collaboration diagram belongs to a class in the class diagram.

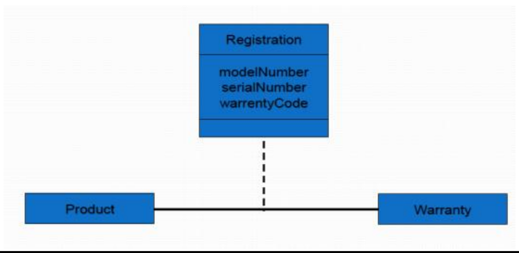
Concept of reflexive relationship

- A class can have a *self association.*


reflexive relationship
Class
Airline Staff
a staff working in an airport may be a pilot, aviation engineer, a ticket dispatcher, a guard, or a maintenance crew member.

Concept of association class

- Associations can also be objects themselves, called *link classes or an association classes.*



Event Tracing

- **For every message shown in a sequence or collaboration diagram,**
  - verify that an operation on the sending class is responsible for sending the event and an operation on the receiving class expects the event and handles it.
  - Verify that there is an association or aggregation on the class diagram between the sending and receiving classes.
  - Add the relationship to the class diagram if it is missing.
  - Finally, if a state transition diagram for the class exists, verify that the event is represented on the diagram for the receiving class.
  - This is needed because the diagram shows all the events that a class may receive.

Documentation Review

- Each class should be documented!
- Check for uniqueness of class names and review all definitions for completeness.
- Ensure that all attributes and operations have a complete definition.
- Finally, check that all standards, format specifications, and content rules/ standards established for the project have been followed.

CRC cards

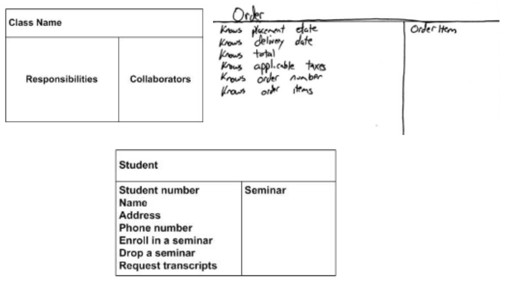
- **Class-responsibility-collaboration** (**CRC**) **cards** are a brainstorming tool used in the design of object oriented software.
- Martin Fowler has described CRC cards as a viable alternative to UML sequence diagram to design the dynamics of object interaction and collaboration.
- CRC cards are usually created from index cards. Members of a brainstorming session will write up one CRC card for each relevant class/object of their design.
- The card is partitioned into three areas:
  - On top of the card, the **class** name
  - On the left, the **responsibilities** of the class
  - On the right, **collaborators** (other classes) with which this class interacts to fulfill its responsibilities

CRC cards

- Using a small card keeps the complexity of the design at a minimum.
- It focuses designers on the essentials of the class and prevents them from getting into its details and implementation at a time when such detail is probably counter-productive.
- It also discourages giving the class too many responsibilities. Because the cards are portable, they can easily be laid out on a table and re-arranged while discussing a design.

CRC Card Index

CRC CARD :



END OF TOPIC 13

-COMING UP!!!!!!

-Model View
-Implementation Diagrams
-Midterm Exam II