

Tutorial

Getting Started

Definitions

- Object- Its an instance of a class.
- Instance- Its an object that behaves in a manner specified by a class.
- Inheritance- Its the ability of one class to define the behavior and data structure of its instances as a superset of the definition of another class or classes.
- Responsibility- Its the knowledge an object maintains and the actions it can perform.
- Server- Its the object that receives the request in a collaboration and thereupon provides the service.
- Collaborator- It represents a server that fulfills a client responsibility.
- Abstract Class- Its a class that cannot be instantiated. It can have at least one virtual operation whose implementation has been deferred to a derived class.
- Attributes- They are variables that are assigned to a class that help fulfill a responsibility.
- Superclass- Its a class from which specific behavior is inherited.
- Subclass- Its a class that inherits behavior from another class.

Choosing the Group

When conducting a CRC session, a group of participants is needed. Selecting a group should be carefully made. Characteristics of a good group can be narrowed down to three categories: size, participants, and dynamics.

The group size for a CRC session should be around five or six participants. Larger groups tend to have differences in views among the participants. Also the number of disagreements increases because the number of potential paths of communication increases as the number of participants grow.

Two kind of participants should attend every CRC session. There are a domain expert and a object-oriented technology facilitator. Also there is a question of whether allowing a manager to attend, but for the tutorial, we will exclude the manager from our CRC session.

The domain expert should understand what the system should do and accomplish. Also he or she should help create the different scenarios and to choose what classes are problem-related. A good person for the domain expert is the author of the requirements document.

The object-oriented technology facilitator should have more experience than others in the group about the object-oriented approach. He or she can aid the CRC session by choosing good classes with meaningful names and minimal but sufficient set of responsibilities. The facilitator should be domain ignorant and help bring up some insightful questions about the application.

The group dynamics should bring together participants who have common goals and agendas, work well together, not dominated by one or few of the members, and respect each other.

Usually the CRC session will be doomed if the group dynamics are not good. If the participants do not agree and communicate poorly, the session is bound for failure.

Before the Scenario Execution

1. The Problem

In defining the problem for the CRC session, a subset of the actual problem should be chosen. This helps the CRC session focus on a fairly small and manageable portion of the entire problem.

This partitioning of the problem is important even in object-oriented design. A object-oriented system can be seen a set of subsystems which then interface through objects of an interface class. This analysis and design is best done one subsystem at a time.

Then the requirements of the system is mentioned again to every participant so that no one is lost or unclear about them. This should take only a few minutes because every participant should have read the requirements prior to the CRC session.

2. Brainstorming for Classes

This stage is when every participant starts spitting off class names that he or she thinks is part of the system. This process consists of a person standing at a board writing the names of the classes as they are suggested by the participants. No discussion should be made about the classes being mentioned are to be valid classes in the system.

Classes can be derived from the requirements document. This practice helps ensure that all known requirements are considered when deciding class names.

3. Filtering Classes

The classes are then examined more closely. Everyone should use the requirements plus knowledge of the participants to rid of redundancies, and recognize related classes.

Also everybody should understand what is meant by each class when deciding precise, and short abstractions for the classes. These abstractions will eventually be entered on the back of the CRC cards.

If no one understands what is meant by a certain class or classes, the domain expert should decide what is the meaning of the class or classes. Everyone should agree on the final classes that are on the board. These classes will turn out to be CRC cards soon.

4. Assigning Cards

After the filtering stage has been completed to its fullest, each class becomes a CRC card. The classes should be assigned to the participants who have a better knowledge of how that class will fit into the system or suggested it during the brainstorming phase. Also combining classes that are similar and assigning them to a participant is helpful and a good idea.

Each participant takes a index card for each class they were assigned. He or she will write a short description of the class on the back of each index card (the side without lines). Each participant then reads their class descriptions to the group for approval and understanding.

Many times prior to a CRC session, the group tries to identify the subclasses and superclasses, responsibilities, and attributes of each class. This can be a good idea only if the relationships are obvious. For the tutorial, we will let these relationships arise from the scenarios.

The Scenario Execution

This is now the heart of the CRC session. We can now start assigning responsibilities to the classes by simulating a scenario. The scenarios to be explored are the "what happens when"s. For example, "What happens when John Barba returns a video, *Batman Returns*, to Blockbuster?"

The scenarios should be very specific. Also related scenarios should be modeled separately. It might seem tedious to model similar scenarios but many of the necessary responsibilities already exist to fulfill it. The group should always execute easy and nonexceptional scenarios first. Only when there is a working model of the system should exceptional scenarios be introduced.

The people who hold and own particular cards should hold the card in the air and "become" the object when a scenario causes control to pass to them. When executing scenarios, cards are lifted to simulate the behavior of the system where messages are sent to actual objects performing their tasks. A card is a class when on the table and an object when lifted in the air.

During the scenarios, responsibilities will arise and should be assigned to the necessary card or cards. Also the collaborators should be written down when responsibilities are known. There may be a time when superclasses or subclasses as well as new classes during the scenario execution.

There are no right and wrong ways for scenario executions. Groups should do what makes the CRC session flow good for them.

During the Scenario Execution

1. Grouping the Cards

Once the group has executed several scenarios to make them feel that their model holds up, the CRC cards should be placed on a table which will provide a visual representation of the model.

Hierarchies of classes should be stacked on top of each other or placed very close together. Classes that collaborate with other classes should be close together as well.

Card clustering can now be seen by the grouping of cards. Classes that seem to be doing too much work or too little can be evaluated by the group for further action.

2. Scenario List

All the scenarios should be recorded as they are suggested and executed. A group can anytime want to reexecute a scenario to recheck a new design or model. Also the list can help see which types of scenarios were not executed and to see which way the model can be tested to fulfill the system requirements. This list will eventually be used as a basic set of test cases for the system.

3. Collaboration Drawings

For a collaboration drawing, cards should be placed somewhere on board, like a whiteboard or blackboard. Then collaboration lines between the classes can be drawn.

This helps the participants see more clearly the sets of interactions. Also the collaboration drawing can point out errors in the system caused by collaborators.

There even may be classes with inward collaborations but no outward collaborations. These classes that require no knowledge of the application can be classified as abstract classes. See figure 2.1

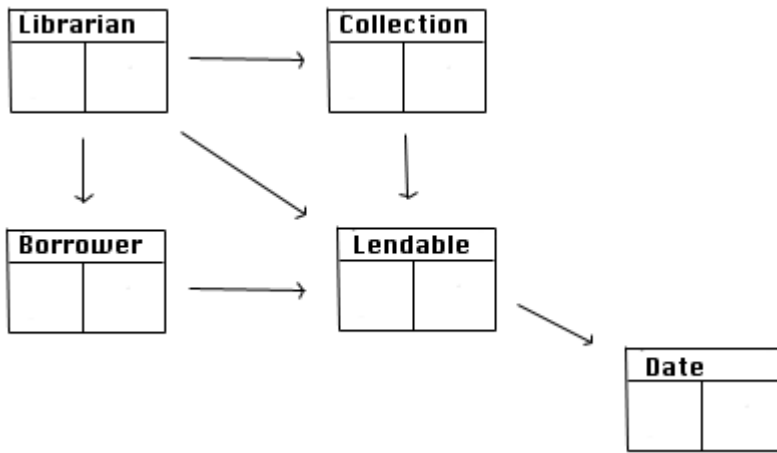


Figure 2.1

After the Scenario Execution

A CRC session can be performed many times on the same system. The group should decide when enough CRC sessions have been executed. Once a stable model is produced, then the group should begin analyzing the model and implementation.

For further record keeping of the CRC session, a automated CRC card technique tool can be used. I have found one from Rational Software Corporation call Rational CRC. This tool should only used as a supplement to the CRC session and not a substitute.

There is a free demo version of Rational CRC at the Rational Software Corporation homepage. If you look in the references section, you will find a link to their page and the setup for the demo is in the System Requirements section. Good luck.