Object Interaction

Object Diagrams

Object collaboration using CRC cards

Object collaboration using a Sequence Diagram

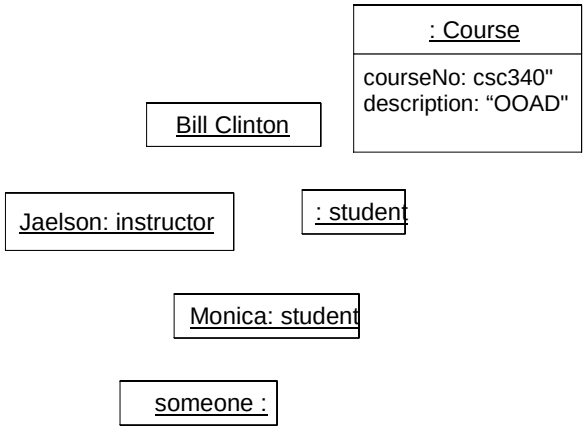
Object collaboration using a Collaboration diagram

How to cross-check between Interaction/Collaboration diagrams and a Class Diagram

Object Diagrams

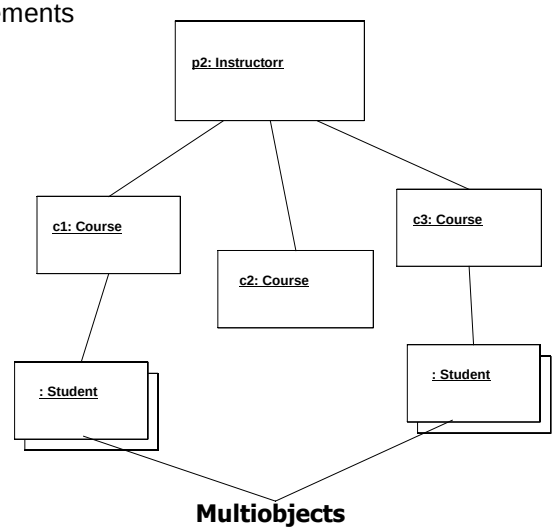
- Model the instances of things contained in class diagrams.
- Show a set of objects and their relationships at a point in time.
- Used to model a snapshot of the system at a moment in time and rendering a set of objects, their state, and their relationships.
- Freeze a moment in time.

Simple Objects



Multiobjects

- **Multiobjects**: set of objects, with undefined number of elements



Communication and Collaboration between Objects

- Fundamental to OO
- Which object should be responsible for each part of the overall behavior?
  - ✓ Allocating operations to appropriate classes.
  - ✓ Atribute X Complex tasks
- What collaborations between objects are required to generate the overall application functionality?
  - ✓ Identifying what messages should be sent between objects and the conditions under which they should be sent.

Object Interaction and Collaboration

- Objects only contain data and methods which are relevant to their own *responsibilities*
  - ✓ They don't "know" about other objects' data.
- To get the kind of functionality which is required for a system to work, the objects have to work together
  - ✓ collaborate
- Objects collaborate by sending messages to one another
  - ✓ calling operations of the required instance
- Objects can only send messages to one another if there is an association between them
  - ✓ Therefore, there must be an association between their classes

Responsibilities

- One way of viewing the problem of assigning operations is in terms of the *responsibilities* of the objects.
- A responsibility is high level description of something a class can do.
  - ✓ Reflects the knowledge or information that is available to that class, either stored within its own attribute or requested via collaboration with other classes.

Responsibilities

- The aim of OO-Analysis-and-Design is to distribute system functionality evenly among its classes
  - ✓ Not that all classes have exactly equal levels of responsibilities, but rather that each class should have appropriate responsibility
- When responsibilities are evenly distributed, each class tends not to be unduly complex
  - ✓ Easier to develop, to test and maintain
  - ✓ Resilient to change in its requirements
  - ✓ A class that it relatively small and self-contained has much greater potential for reuse

CRC (Class Responsibility Collaboration) Cards

- One way to assign operations to objects is to work with CRC Cards to determine the responsibilities of each object.
- CRC stands for:
  - ✓ Class
  - ✓ Responsibility
  - ✓ Collaboration
- CRC can be used at several different stages of project, for different purposes
  - ✓ Here we use it in modeling object interaction

CRC Cards

- These are cards which for each class show what its responsibilities are, in terms of
  - ✓ its attributes
  - ✓ the operations it can carry out
  - ✓ the names of the other classes it needs to collaborate with in order to carry out its responsibilities

Role Play with CRC Cards

- In analysis we can spend time role playing with CRC cards to try to sort out the responsibilities of objects and to determine which are the other objects they need to collaborate with in order to carry out those responsibilities
- Often the responsibilities start out being vague and not as precise as the operations which may only become clear as we move into design.
- Sometimes we need to role play the objects in the system and test out the interactions between them.

I'm a Campaign ....

"I'm a Campaign. I know my title, start date, finish date and how much I am estimated to cost.

"When I've been completed, I know how much I actually cost and when I was completed. I can calculate the difference between my actual and estimated costs.

"When I've been paid for, I know when the payment was made.

"I can calculate the contribution made to me by each member of staff who worked on me."

I'm a CreativeStaff ....

"I'm a CreativeStaff. I know my staff no, name, start date and qualification.

"I can calculate how much bonus I am entitled to at the end of the year."

Does it make sense to include

"I can calculate the contribution made to each campaign I have worked on by each member of staff who worked on it."

or does that belong in Campaign?

Class: *Campaign*	
Responsibilities:	**Collaborating Classes**
Title	
StartDate	
FinishDate	
EstimatedCost	
ActualCost	
CompletionDate	
DatePaid	
AssignManager	*CreativeStaff*
RecordPayment	
Completed	
GetCampaignContribution	
CostDifference	

Class: *CreativeStaff*	
Responsibilities:	**Collaborating Classes**
StaffNo	
StaffName	
StaffStartDate	
Qualification	
CalculateBonus	*Campaign*
ChangeGrade	*StaffGrade*
	Grade

Sequence Diagrams

- CRC cards and role playing may help to find out what the responsibilities of objects are and which other objects they need to collaborate with, but
  - ✓ They **do not** provide a way of **documenting** these **interactions** for a particular Use Case -> Sequence Diagrams
- An **interaction** is a behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose.
- A **message** is a specification of a communication between objects that conveys information with the expectation that activity will ensue.
  - ✓ Message types: sync, async, time-out, uncommitted, etc.

Sequence Diagrams

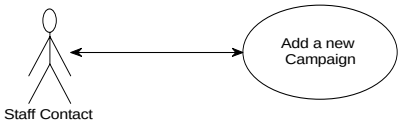
- There will be a Sequence Diagram (SD!) for each Use Case in the system.
- We draw SDs in order to document how objects *collaborate* to produce the functionality of each use case.
- The SDs show the data and messages which pass across the system boundary and the messages being sent from one object to another in order to achieve the overall functionality of the Use Case

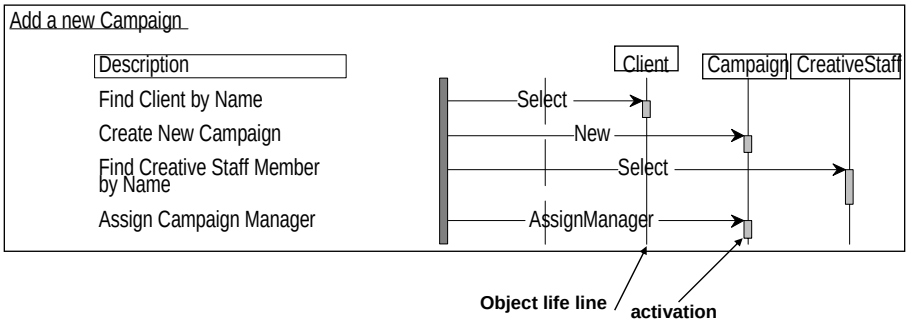
Sequence Diagrams

- A Sequence Diagram can be seen as an expansion of a use case to the lowest possible level of detail.
- Sequence diagrams can be drawn at different levels of details and also to meet different purposes at several stages in the development life-cycle.
- Analysis Sequence Diagrams normally **do not**
  - ✓ include design objects
  - ✓ Specify message in any detail

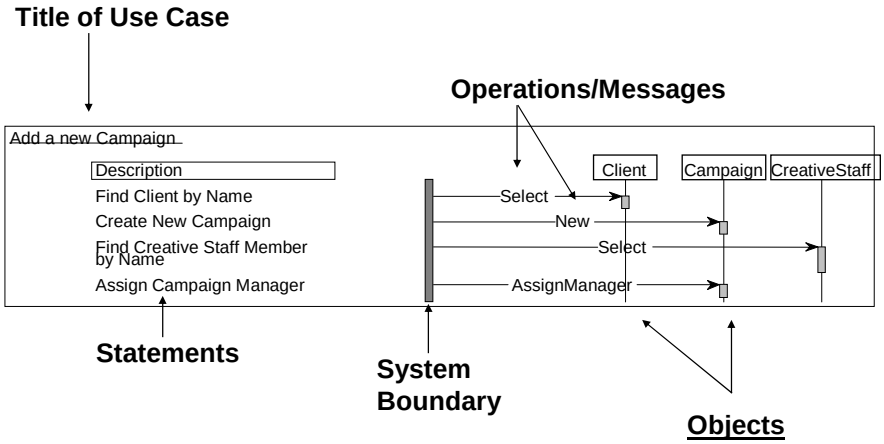
Example: Add a new Campaign

- This SD is for the Use Case “Add a new Campaign”

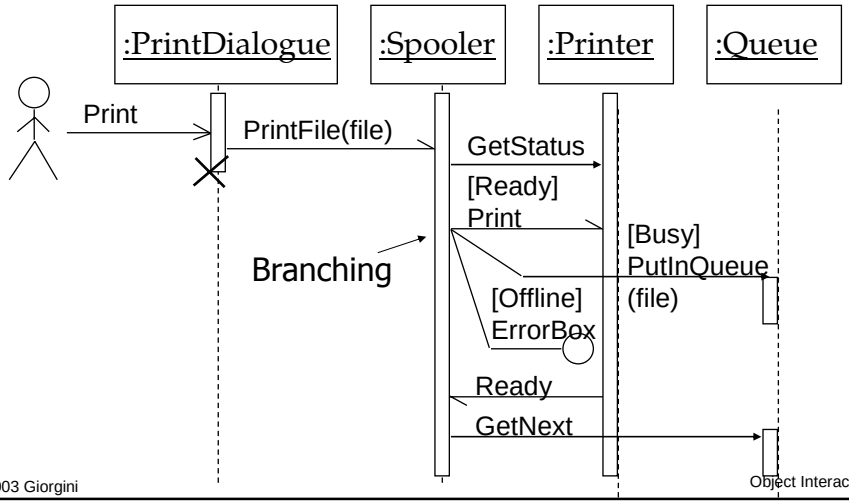




Example: Add a new Campaign



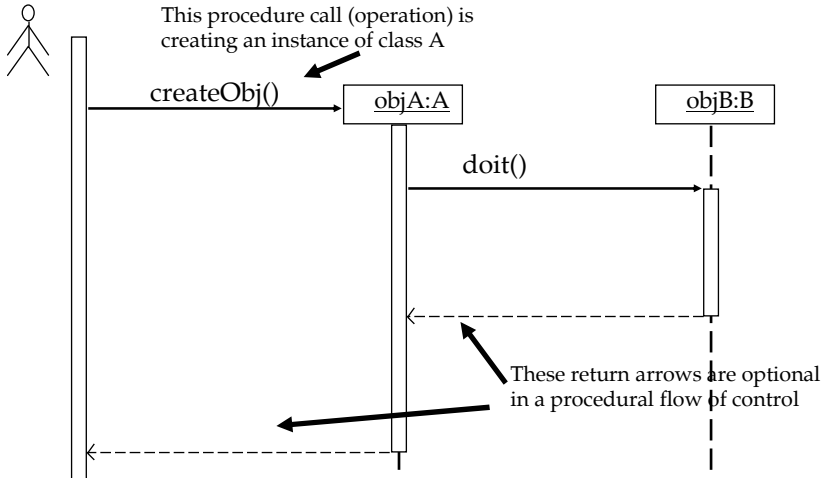
Example: Printing



Notation

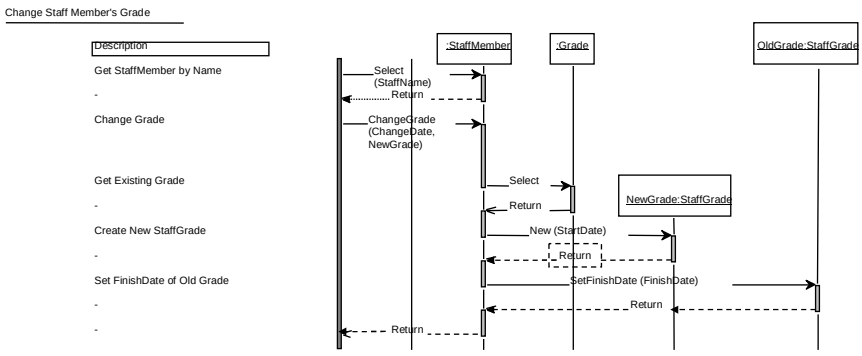
- The name of the class and the name of the instance of that class (object), if required, at the top of the column that represents it.
- Including the arrows marked Return makes it easier to follow the flow of control from object to object
- The vertical line for each object represents its *lifeline*
- The thick bar represents its *activation*
  - ✓ i.e. when it is doing something

Sequence Diagrams: UML



Sequence Diagrams: UML

- The box representing an object is drawn at the top if the object already exists, and at the point where the object is created if it is created within this use case, i.e. at the start of its lifeline



Structures

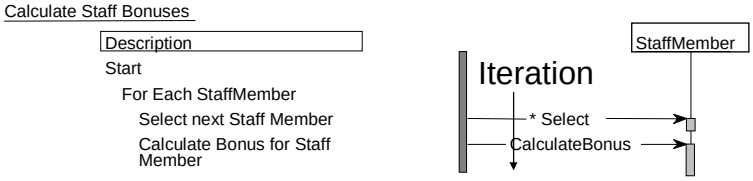
- The statements in SDs can be structured using the three structures which are common to all computer programs:
  - ✓ Sequence - one statement following another;
  - ✓ Selection - a choice between alternatives
    - ✓ if statement
    - ✓ if ... else statement
    - ✓ case statement
  - ✓ Iteration - repeating statement(s) in a loop controlled by a condition
    - ✓ for ... next
    - ✓ while ... do ... loop
    - ✓ until ... do ... loop

Iteration

Iteration (repetition of an operation) is shown with an asterisk

Each `StaffMember` will be selected in turn Once selected, the `CalculateBonus` message will be sent to the one currently selected

There is only one loop!

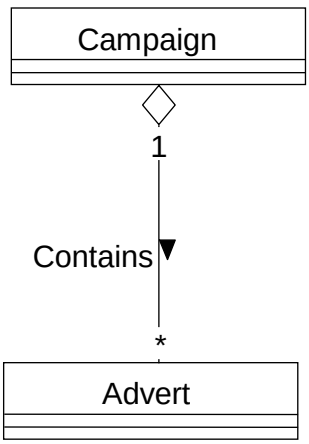


Drawing SDs

- For a particular use case, start by identifying which objects might be involved
- You may not get this right, but you can always change it
- Imagine that there is a use case required by Agate called `Check Campaign Budget`
- Each `Campaign` has an `EstimatedCost` attribute and each `Advert` has an `EstimatedCost` attribute
- The purpose of the use case is to check that the total estimated cost of all the adverts is less than that for the campaign as a whole.
  - ✓ Which objects are involved?

Drawing SDs

Class diagram showing aggregation



Drawing SDs

Class diagram for Campaign class

Campaign
+Title : String +CampaignStartDate : Date +CampaignFinishDate : Date +EstimatedCost : Money +ActualCost : Money +CompletionDate : Date +DatePaid : Date -StaffCount : Integer = 0
+Completed (CompletionDate : Date, ActualCost : Money) +SetFinishDate (FinishDate : Date) +RecordPayment (DatePaid : Date) +CostDifference ():Money +GetCampaignContribution ():Money +CheckBudget ():Money

Drawing SDs

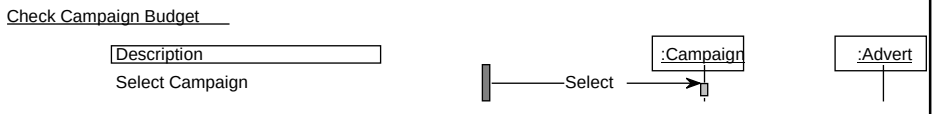
Class diagram for Advert class

Advert
#Title : String #Type : String #TargetDate : Date #CompletedDate : Date #EstimatedCost : Money
+SetCompleted (CompletedDate : Date =Today's Date) +GetTitle () :String +GetType () :String +GetTargetDate () :Date +GetCompletedDate () :Date +GetCost () : Money

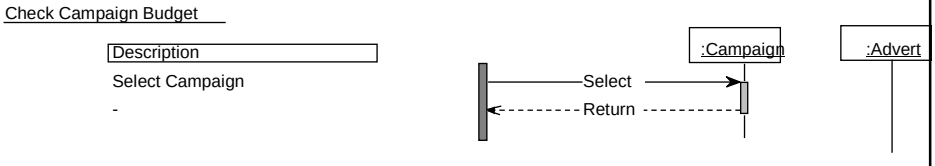
Drawing SDs

- What is the first thing to do?
- Select the relevant Campaign, probably using its name
- How we select it may be something we leave until we get into design:
  - ✓ it could be from a list box
  - ✓ it could involve a separate window on the screen
  - ✓ it could involve some kind of index
- These are design issues, which we shall leave for now, although we would document them if the customer expressed a preference at this stage

Drawing SDs

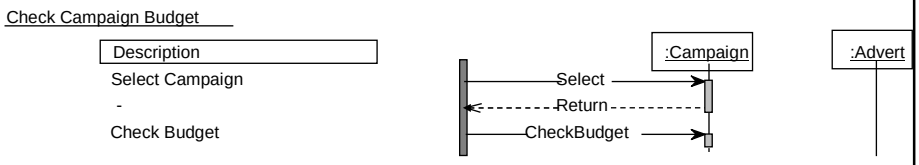


- We can add in a Return



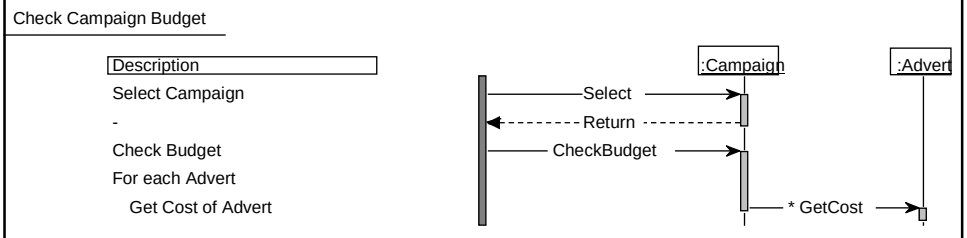
Drawing SDs

- We then need to send a message to the Campaign to check its budget.



- Note there is no Return here. Where does control go?

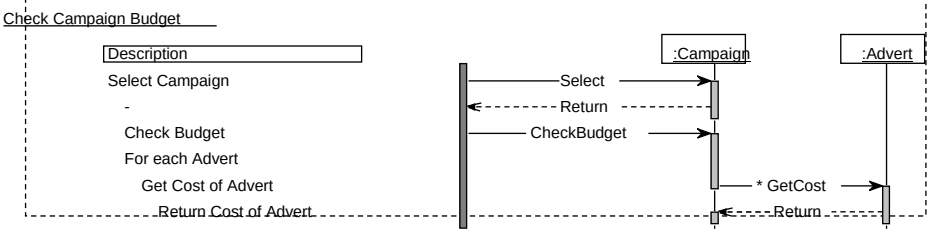
Drawing SDs



- Note the * for iteration.
  - ✓ We are assuming here that the Campaign knows about all the Adverts that are contained in it by means of the Aggregation.

Drawing SDs

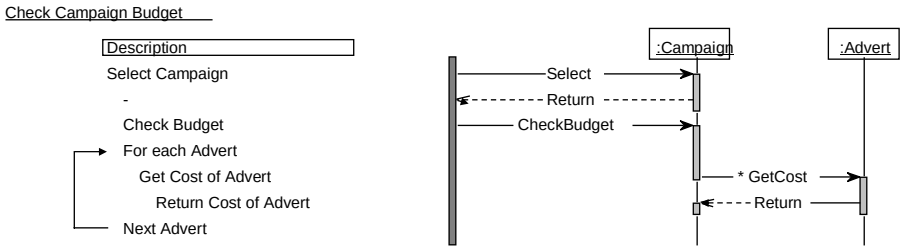
- What happens next?



- Advert returns its cost, in this case the EstimatedCost of the Advert

Drawing SDs

- This has to happen for every Advert in the Campaign, so there's a loop

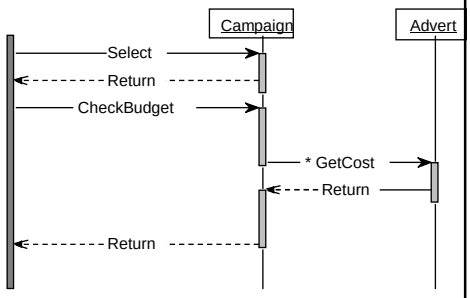


- Once all the Adverts' costs have been fetched and totalled up, the total can be taken away from the EstimatedCost of the Campaign.

Drawing SDs

Check Campaign Budget

Description
Select Campaign
-
Check Budget
For each Advert
Get Cost of Advert
Return Cost of Advert
Next Advert
Return (Estimated Cost - Cost of Adverts)



- If it is negative, then the Campaign is likely to exceed its budget

Relationship to Class Diagram

Advert
#Title : String #Type : String #TargetDate : Date #CompletedDate : Date #EstimatedCost : Money #ActualCost : Money
+SetCompleted (CompletedDate : Date =Today's Date) +GetTitle () :String +GetType () :String +GetTargetDate () :Date +GetCompletedDate () :Date +GetCost () : Money

- We could add a new attribute to Advert called ActualCost, which is set when the Advert is completed
- Now GetCost can return the ActualCost if it exists, otherwise the EstimatedCost

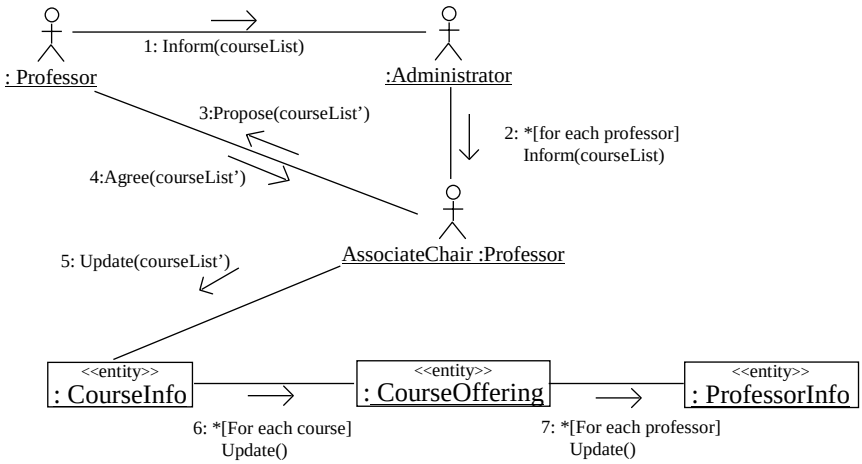
Collaboration Diagram

- Emphasizes the organization of the objects that participate in an interaction.
- Shows interaction without the time dimension but do include object links.
- Collaboration diagrams are intended to model scenaria; each scenario describes a possible sequence of events and actions.
- For complex use cases use several collaboration diagrams; make sure each collaboration diagram is simple and easy to understand.

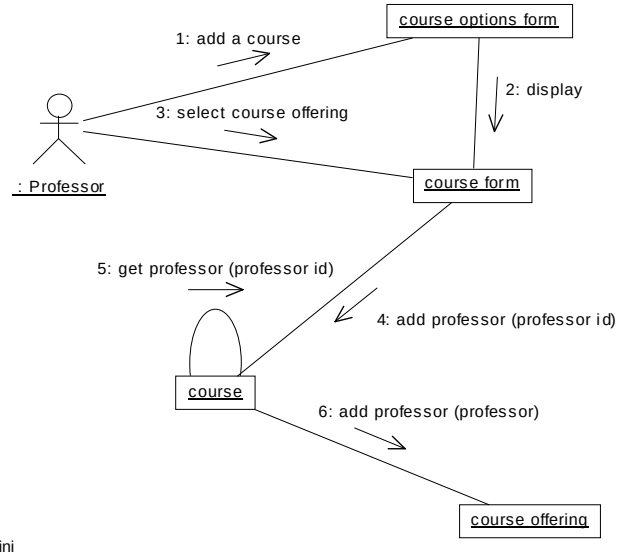
Collaboration Diagram

- The interaction starts from an actor.
- Links between objects are shown and may indicate the direction of navigability with arrowheads.
- Messages are shown beside the links with a directional arrow.
- Messages are numbered to show their sequence.

Ex: Select Courses to Teach



Ex: Add course offerring



Sequence Diagram X Collaboration Diagram

- Sequence diagrams and collaboration diagrams are semantically equivalent.
- You can take a diagram in one form and convert it to the other without loss of information.
- To model flows of control by time ordering (Sequence diagram).
  - ✓ Emphasizes the passing of messages as they unfold over time -> visualization of Use Case scenario.
- To model flows of control by organization (Collaboration Diagram).
  - ✓ Emphasizes the structural relationships among instances in the interaction -> visualization of multiple concurrent flows of control

Relationship to Class Diagram

- All operations shown on the collaboration and sequence diagrams must be present in the destination classes on the class diagram.
- Links shown on the collaboration diagram normally correspond to associations on the class diagram.

Additional Readings

- [Booch99] Booch, G. et all. The Unified Modeling Language User Guide. Chapters 15, 18, 27. Addison-Wesley.
- [Jacobson92] Jacobson, I. et all. Object-Oriented Software Engineering: A Use-Case Driven Approach. Addison-Wesley.
- [Fowler00] Fowler, M. et all. UML Distilled: A Brief Guide to the Standard Object Modelling Language. Chapter 5. Addison-Wesley.