IX. Sequence and Collaboration Diagrams

Interaction Diagrams
Sequence Diagrams
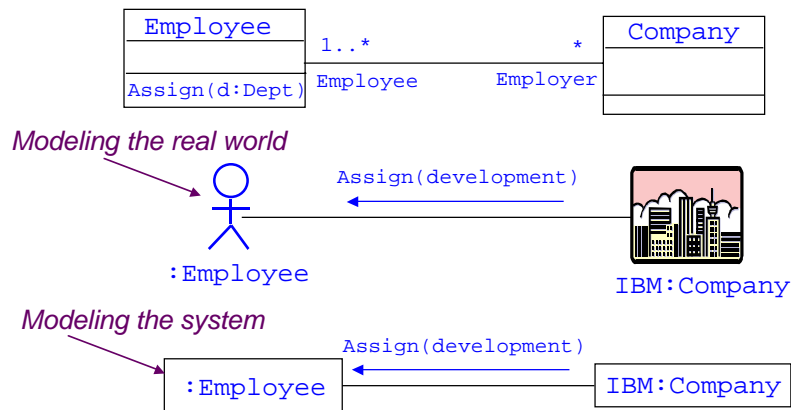**Examples**
Collaboration Diagrams

Interaction Diagrams

- *Interactions among objects are modeled by **interaction diagrams.***
- *An interaction between two objects A and B involves object A sending a message requesting an action that object B can perform.*
- *There are two types of interaction diagrams:*
    - ✓ *Sequence diagrams;*
    - ✓ *Collaboration diagrams.*
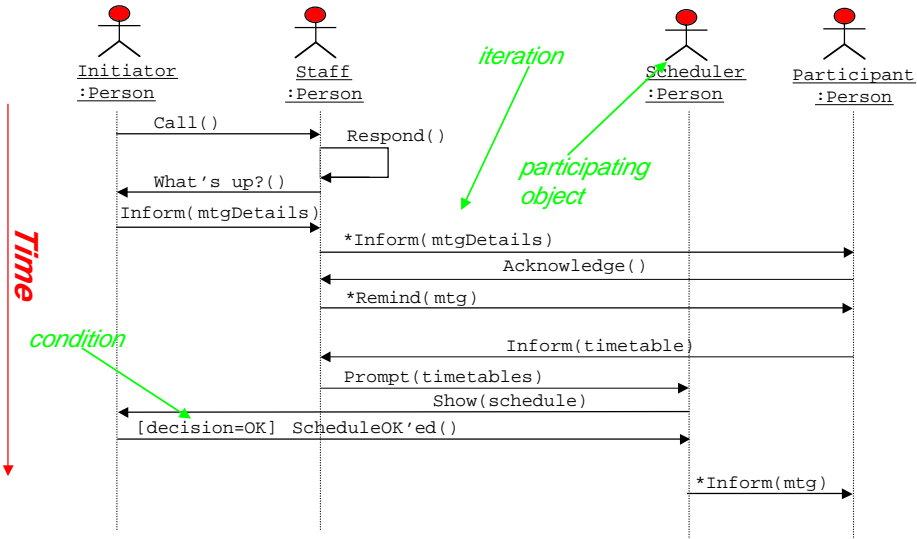- *We discuss each in detail in the rest of this lecture unit.*

The Nature of an Interaction

Employee	1..*	*	Company
Assign(d:Dept)	Employee	Employer	

Modeling the real world

Assign(development)

:Employee   IBM:Company

Modeling the system

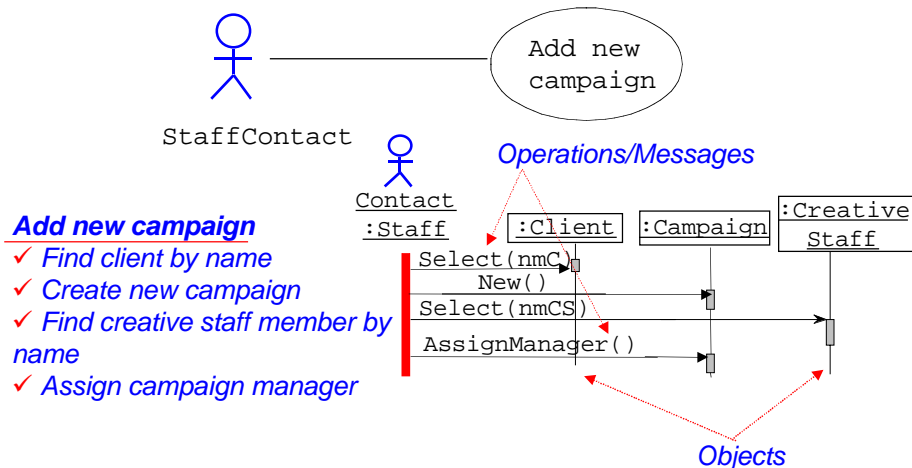:Employee ← Assign(development) ← IBM:Company

---

Sequence Diagrams

- *Sequence diagrams describe in detail how actors use use cases; they can also model external business processes the new system will support (e.g., processing a book order)*
- *An **interaction** is a behavior that consists of a set of messages exchanged between external and system objects.*
- *Interactions consist of one or more **messages**. Interactions may be synchronous (e.g., calling someone on the phone), or asynchronous (e.g., sending someone email).*
- *Sequence diagrams defined during requirements analysis should **not**:*
  - ✓ *include design objects;*
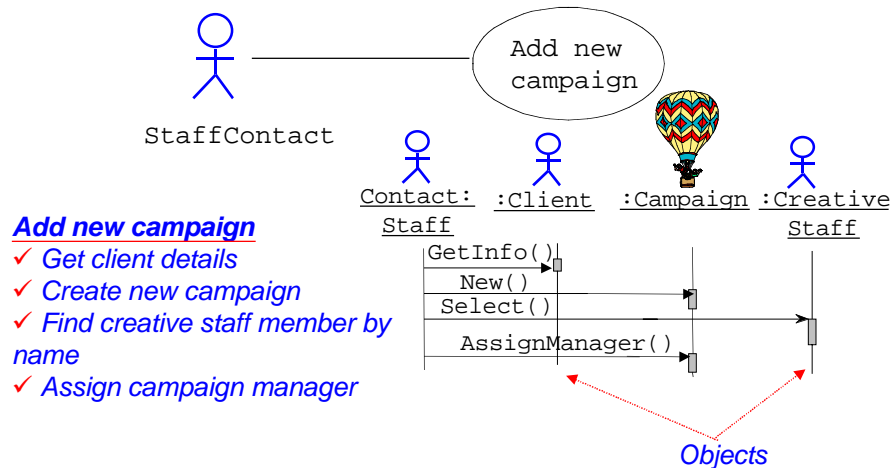 - ✓ *specify message signatures in any detail;*

The Basic Idea

Initiator
:Person

Staff
:Person

iteration

Scheduler
:Person

Participant
:Person

*participating
object*

Time

Call()

Respond()

What's up?()

Inform(mtgDetails)

*Inform(mtgDetails)

Acknowledge()

*Remind(mtg)

condition

Inform(timetable)

Prompt(timetables)

Show(schedule)

[decision=OK] ScheduleOK'ed()

*Inform(mtg)

Example: Add a New Campaign

- *Getting back to the use case "Add a new campaign"*

Add new
campaign

StaffContact

Contact
:Staff

Operations/Messages

:Client

:Campaign

:Creative
Staff

Add new campaign
✓ Find client by name
✓ Create new campaign
✓ Find creative staff member by name
✓ Assign campaign manager

Select(nmC)

New()

Select(nmCS)

AssignManager()

Objects

Add another New Campaign

■ *Getting back to the use case "Add a new campaign"*

StaffContact

Add new campaign

Add new campaign
- ✓ Get client details
- ✓ Create new campaign
- ✓ Find creative staff member by name
- ✓ Assign campaign manager

Contact: Staff   :Client   :Campaign   :Creative Staff

GetInfo()
New()
Select()
AssignManager()

Objects

A More Realistic Example

Add new campaign
- ✓ Find client by name;
- ✓ Create new campaign c;
- ✓ Assign creative staff member to c;
- ✓ Assign campaign manager;
- ✓ Inform the creative staff person.

Staff Contact: :Client   *New object*   :Creative Staff   :Creative StaffE

c:Campaign

Select(nm)
New(nm)
Select(c)
AssignManager()
Inform(c)

An Even More Realistic Example

External objects

New object

Staff
Contact: | :Client |

Select(nm)
New(nm)

c:Campaign

Select(c)
AssignManager() c:CampaignE

Inform(c)

:Creative
Staff

:Creative
StaffE
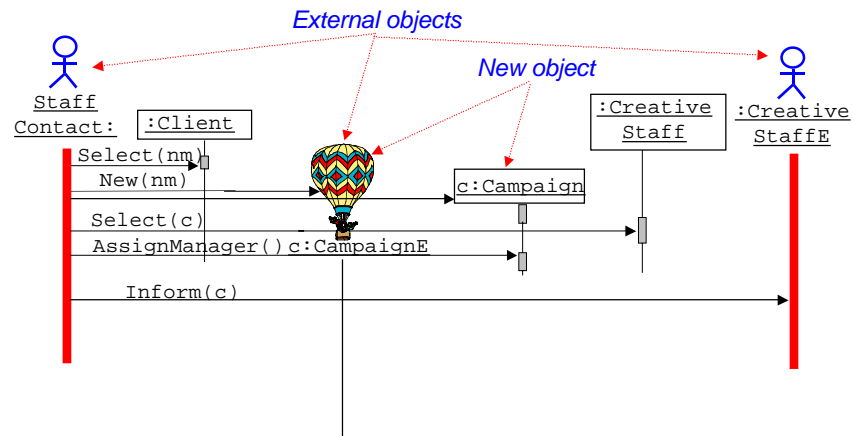
Another Example: Print Shop

:CustomerP   :PrinterP | :Printer | | :Queue |

Lifeline

PrintFile(file)

GetStatus()   *Inactive*

[Ready]Print()   *Active*

[Busy]
PutInQueue
(file)

Branching

[OutOfService]
CallRepair

Done

Ready(file)

Ready(file)

GetNext()

Asynchronous

Flow of Control

This procedure call (operation) is
creating an instance of class A

createObj()

objA:A   objB:B

doit()

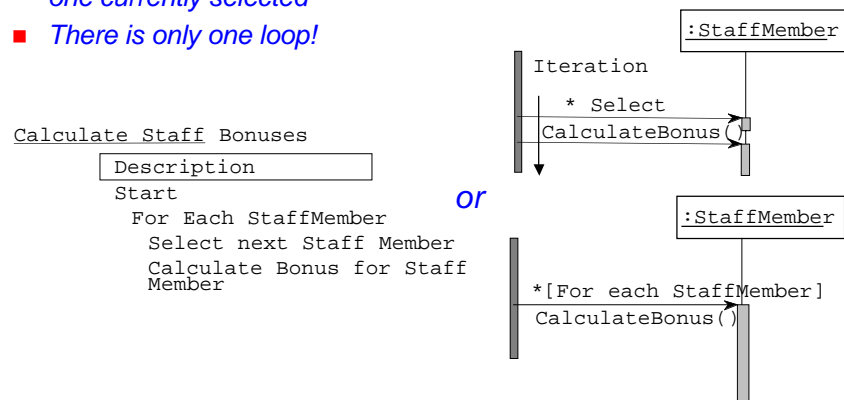*These return arrows are optional*
in a procedural flow of control
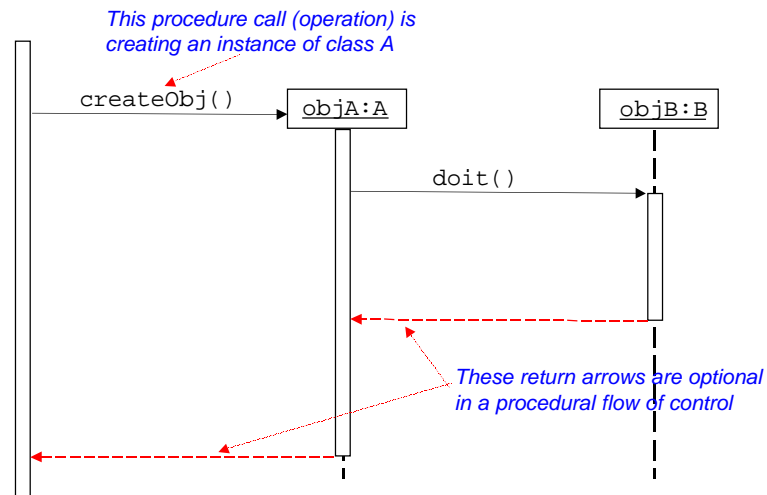
Iteration

- *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!*

:StaffMember

Iteration

* Select

Calculate Staff Bonuses

CalculateBonus()

Description

Start   *or*
  For Each StaffMember
    Select next Staff Member
    Calculate Bonus for Staff
    Member

:StaffMember

*[For each StaffMember]
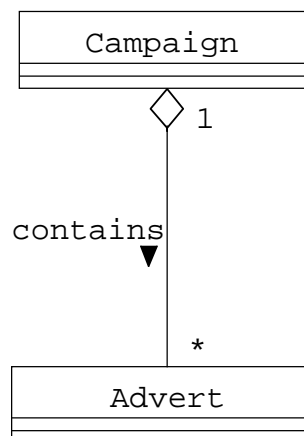CalculateBonus()

Drawing Sequence Diagrams

- *For a particular use case, start by identifying which objects and actors 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 here?*

`Campaign` *and* `Advert`

Class diagram showing aggregation

```
┌─────────────────────┐
│      Campaign        │
├─────────────────────┤
├─────────────────────┤
└─────────────────────┘
           ◇ 1
           │
contains   │
           ▼
           │
           * 
┌─────────────────────┐
│       Advert         │
├─────────────────────┤
├─────────────────────┤
└─────────────────────┘
```

The `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

---

The `Advert` Class

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

Getting a Sequence Diagram

- *Where do we start?*
- *Select the relevant* `Campaign`*, probably using its name.*
- *How we select it is something we leave for the design phase:*
  - ✓ *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 should document them if the customer expressed a preference at this stage.*
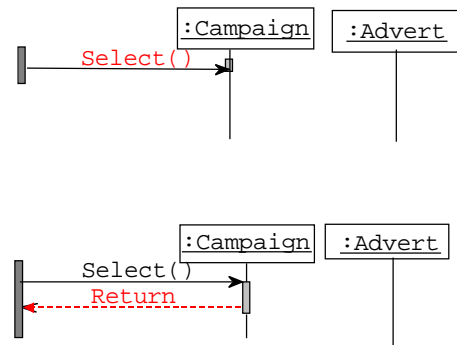
Creating a Sequence Diagram

Check Campaign Budget

Description
Select Campaign

:Campaign   :Advert

Select()

- *We can also add in a* `Return`

:Campaign   :Advert

Select()

Return

Creating a Sequence Diagram

- ■ *We then need to send a message to the* `Campaign` *to check its budget.*
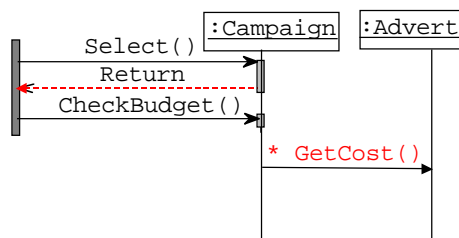
Check Campaign Budget

Description
Select Campaign
-
Check Budget

`:Campaign` `:Advert`

Select()
Return
CheckBudget()
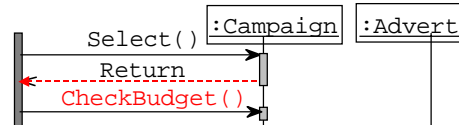
- ■ *Note there is no* `Return` *here. Where does control go?*

Creating a Sequence Diagram

Check Campaign Budget

Description
Select Campaign
-
Check Budget
For each Advert
Get Cost of Advert

`:Campaign` `:Advert`

Select()
Return
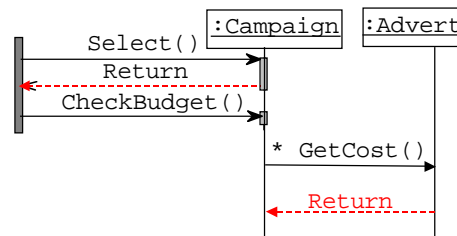CheckBudget()
* GetCost()

- ■ *Note the * for iteration.*
- ■ *We are assuming here that* `:Campaign` *knows about all the* `Advert`*s that are contained in it because of the aggregation association shown earlier.*

Creating a Sequence Diagram

■ *What happens next?*

```
Check Campaign Budget
    Description
    Select Campaign
    -
    Check Budget
   For each Advert
    Get Cost of Advert
      Return Cost of Advert
```
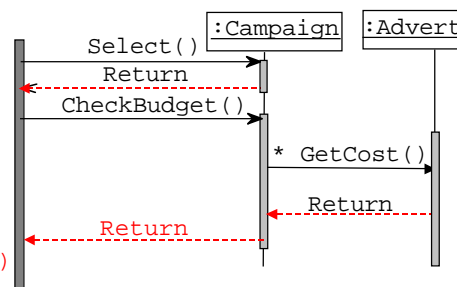

■ Advert *returns its cost, in this case the* EstimatedCost *of the* Advert

■ *Once all the* Advert's *costs have been fetched and totalled up, the total can be taken away from the* EstimatedCost *of the* Campaign*.*

Creating a Sequence Diagram

■ *This has to happen for every* Advert *in the* Campaign, *so there's a loop*
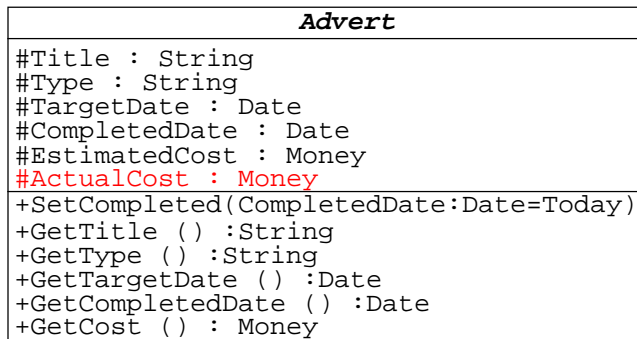
```
Check Campaign Budget
    Description
    Select Campaign
    -
    Check Budget
   For each Advert
    Get Cost of Advert
      Return Cost of Advert
   Return (Estimated Cost
          - Cost of Adverts)
```


■ *Once all the* Advert's *costs have been fetched and totalled up, the total can be taken away from the* EstimatedCost *of the* Campaign*.*

...Back to Class Diagrams...

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

- *We could add a new attribute to* `Advert` *called* `ActualCost`, *which is set when an* `Advert` *has been completed.*
- *Now* `GetCost()` *can return the* `ActualCost` *if it exists, otherwise it uses* `EstimatedCost().`

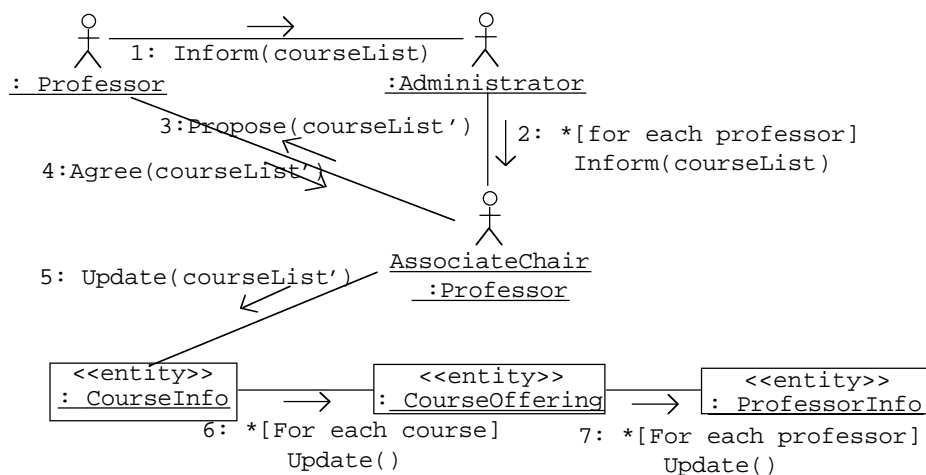# How to Use Sequence Diagrams

- *In general, you may need several sequence diagrams to describe a single use case.*
- *A use case may involve complex control logic; sequence diagrams on the other hand should remain easy to read and understand.*
- *For a complex use case, use several sequence diagrams, each of which describes a possible scenario for the use case.*
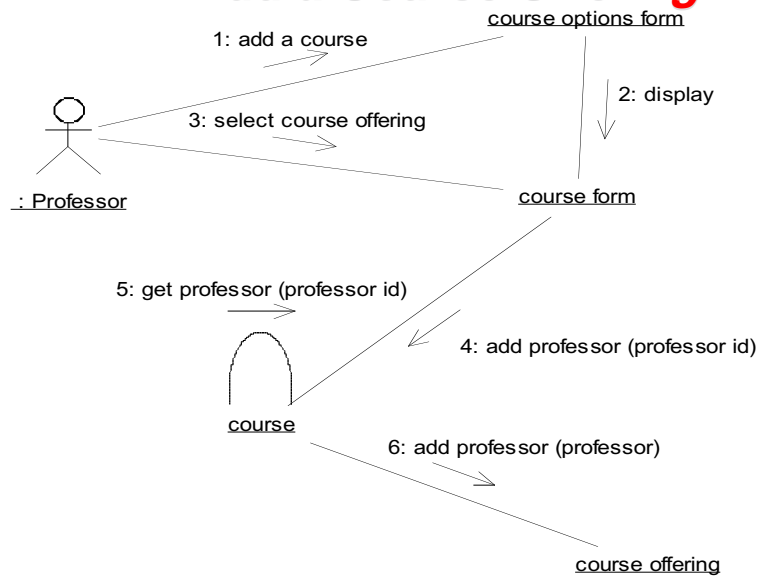
Collaboration Diagrams

- *These diagrams are comparable to sequence diagrams. In fact, you can map every sequence diagram to an equivalent collaboration diagram and vice versa.*
- *Collaboration diagrams show interaction without the time dimension, but do include object links.*
- *Like sequence diagrams, collaboration diagrams are intended to model scenaria; each scenario describes a possible sequence of events and actions.*
- *Sequence diagrams are helpful because they capture visually the sequence of events over time.*
- *Collaboration diagrams capture more directly the interactions between actors and objects.*
- ***Note:*** *All operations shown on collaboration and sequence diagrams must be present in the destination classes.*

Select Courses to Teach

Add a Course Offering

course options form

1: add a course

2: display

3: select course offering

: Professor

course form

5: get professor (professor id)

4: add professor (professor id)

course

6: add professor (professor)

course offering

Additional Readings

- *[Booch99] Booch, G. et al. The Unified Modeling Language User Guide. Chapters 15, 18, 27. Addison-Wesley.*
- *[Fowler00] Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modelling Language. Chapter 5. Addison-Wesley.*