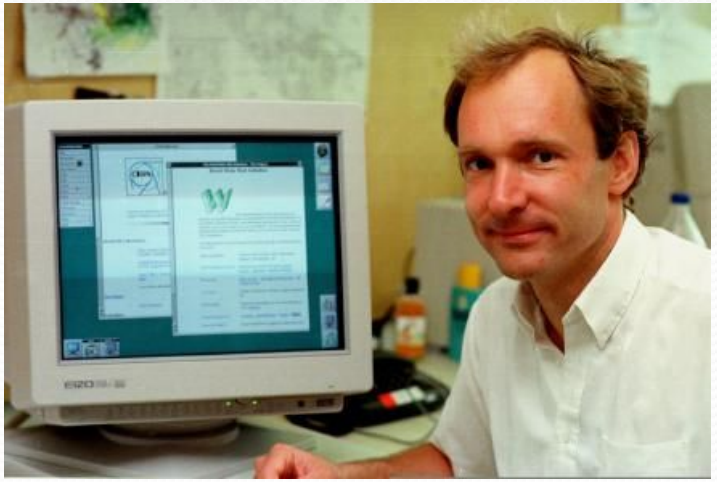
System Modeling

System modeling

- Abstract models of a system
- Ignores system details
- Different perspective of system
- Helps the analyst to understand the **functionality** of the system
- Models are used to **communicate** with customers

Let's remember a timely example

- Proposal for Web (1989) by

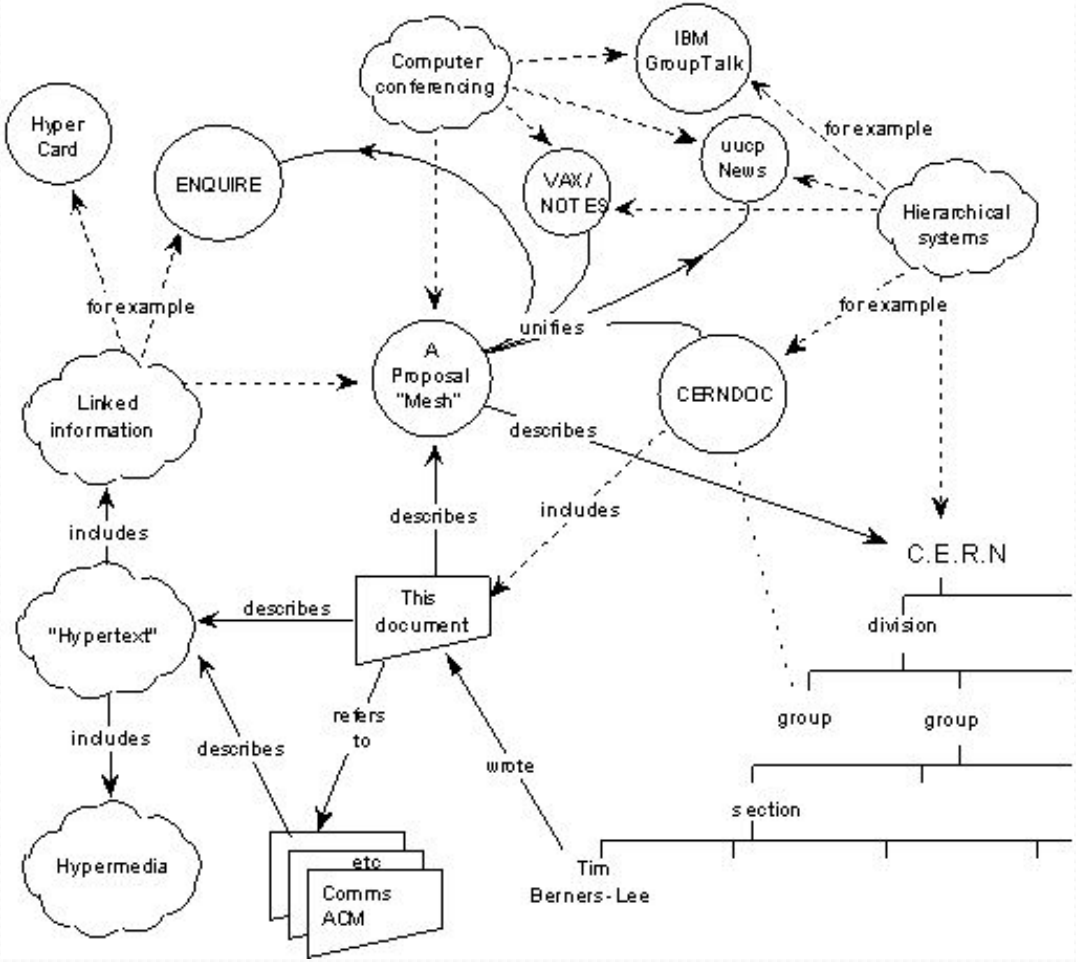


Sir Tim Berners-Lee

History of the Web

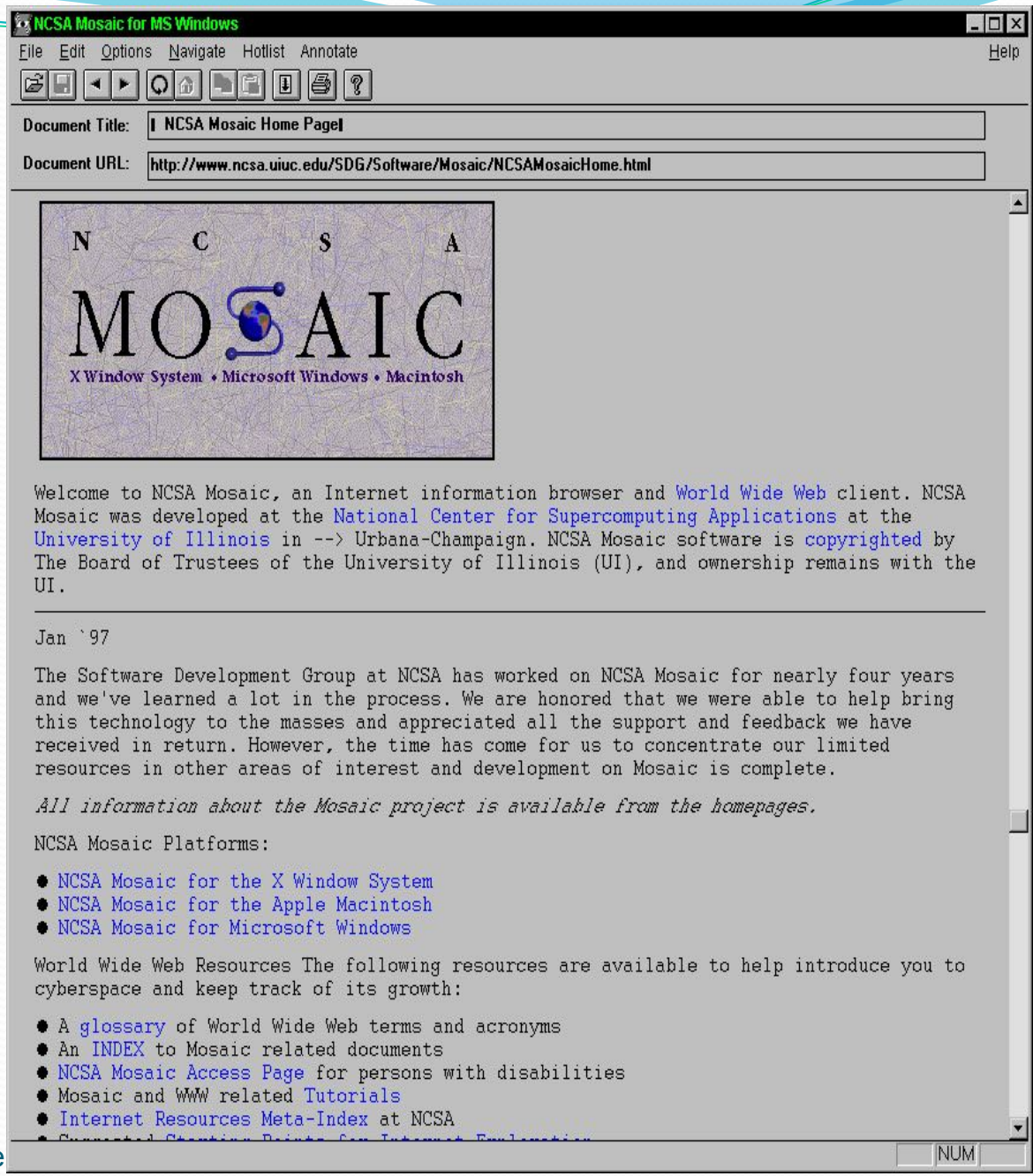
- Big issues:
  - security,
  - surveillance
  - privacy
  - open infrastructure
  - net neutrality
  - content protection
  - …

Proposal for Information Management at CERN by TBL



Written:
March1989
Redistributed:
May 1990

First Graphical Web Browser *NCSA MOSAIC*



System mode

System perspectives

- **External** perspective models
  - the context or environment of the system.
- **Interaction** perspective models between:
  - system and its environment
  - the components of a system
- **Structural** perspective models
  - the **organization** of a system
  - the **structure** of the data
- **Behavioral** perspective models
  - the **dynamic** behavior of the system
  - how it responds to events.

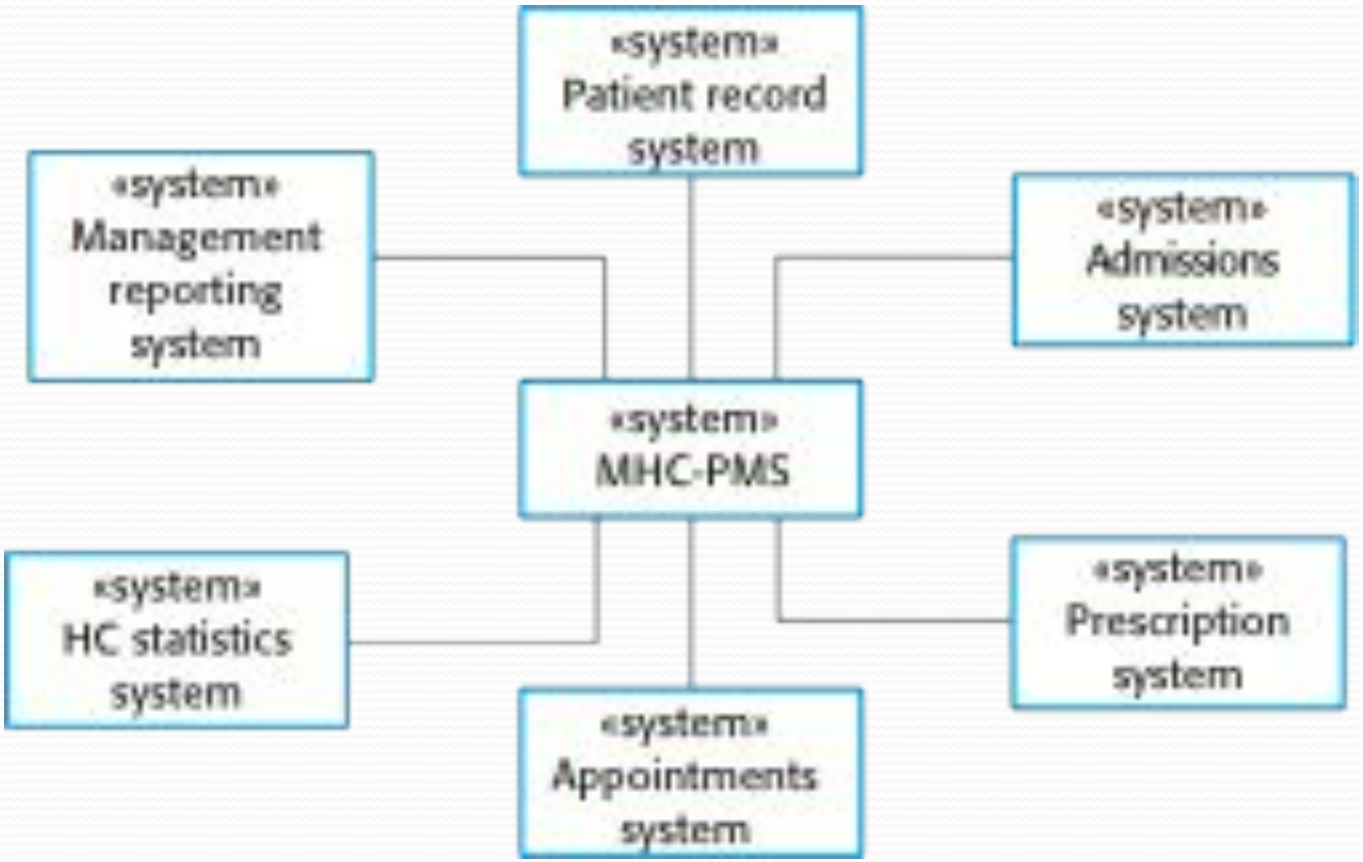
UML diagram types

- **Activity** diagrams
  - **Activities** in a process
- **Use case** diagrams
  - **interactions** between system & environment
- **Sequence** diagrams interactions between
  - **actors** & system
  - system **components**
- **Class** diagrams
  - object classes in the system
  - associations between classes
- **State** diagrams
  - system response to internal and external events

Context models

- Operational context
- Relationship to other system
- Shows **boundaries**
  - what is inside and what is outside
- Boundary setting is important
  - Huge impact on requirements

The context of MHS-PMS

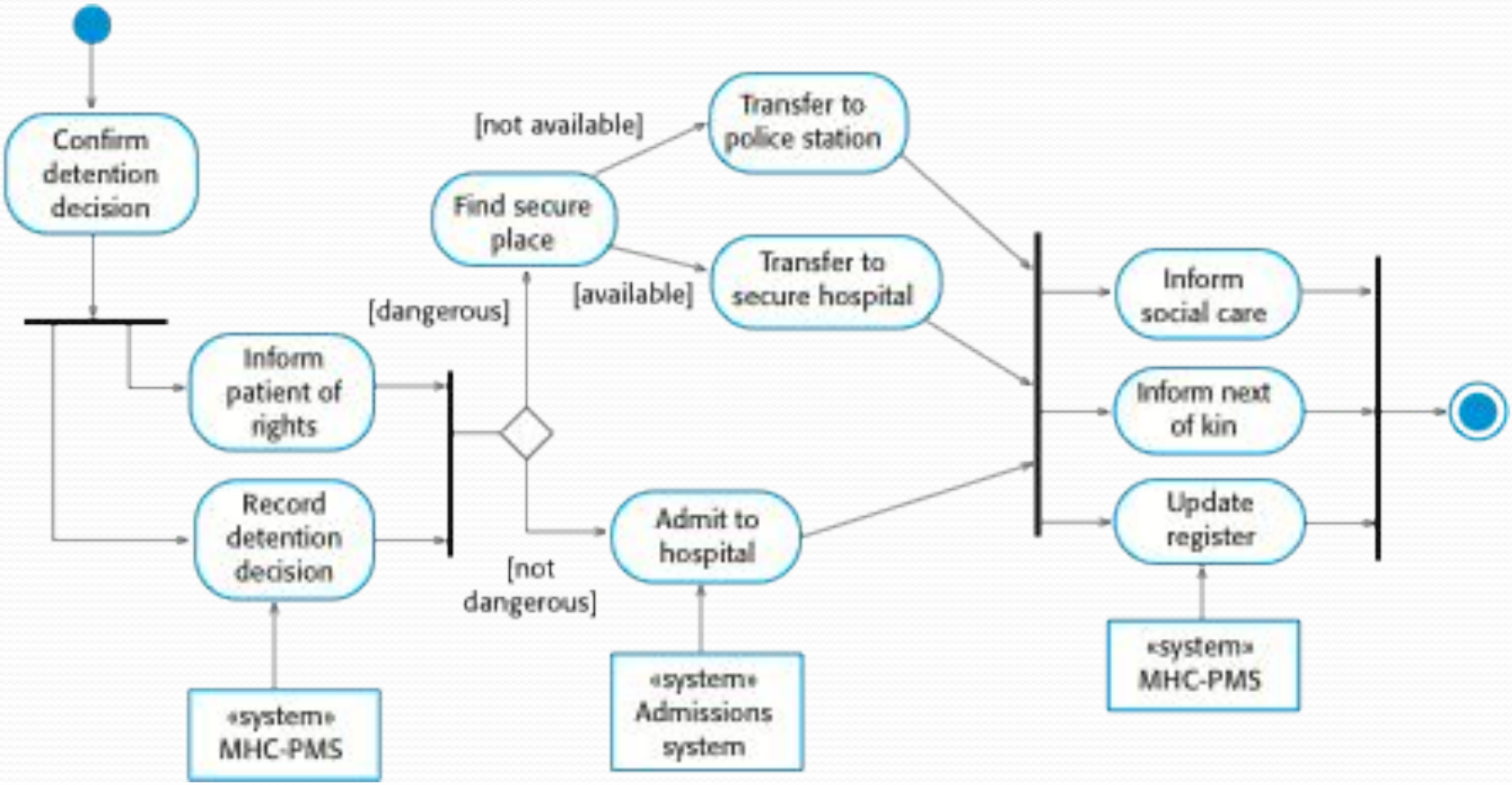


Source: Sommerville Software Engineering Examples

Process perspective

- Shows how the system is used
- UML **activity diagrams**

Process model: involuntary detention



Interaction models

- Use case diagrams
  - User interaction
  - Useful for requirements
  - Communication issues revealed
- Sequence diagrams
  - Component interaction
  - Useful to determine if system will satisfy requirements

Use case modeling

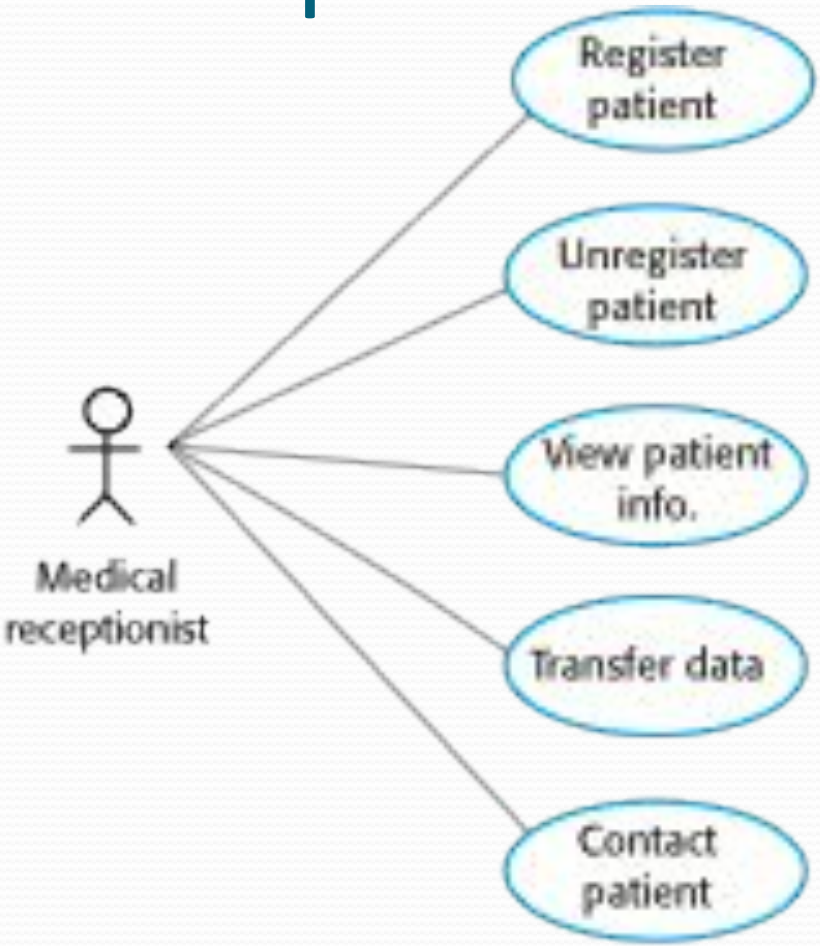
- Each use case represents a discrete task
- Involves external interactions
- Actors
  - People
  - Organizations
  - Other systems

Use case: Transfer Data

Description 'Transfer data' use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Use cases in the MHC-PMS role ‘Medical Receptionist’



Structural models

- **Organization** of a system
  - components of the system
  - relationships between components

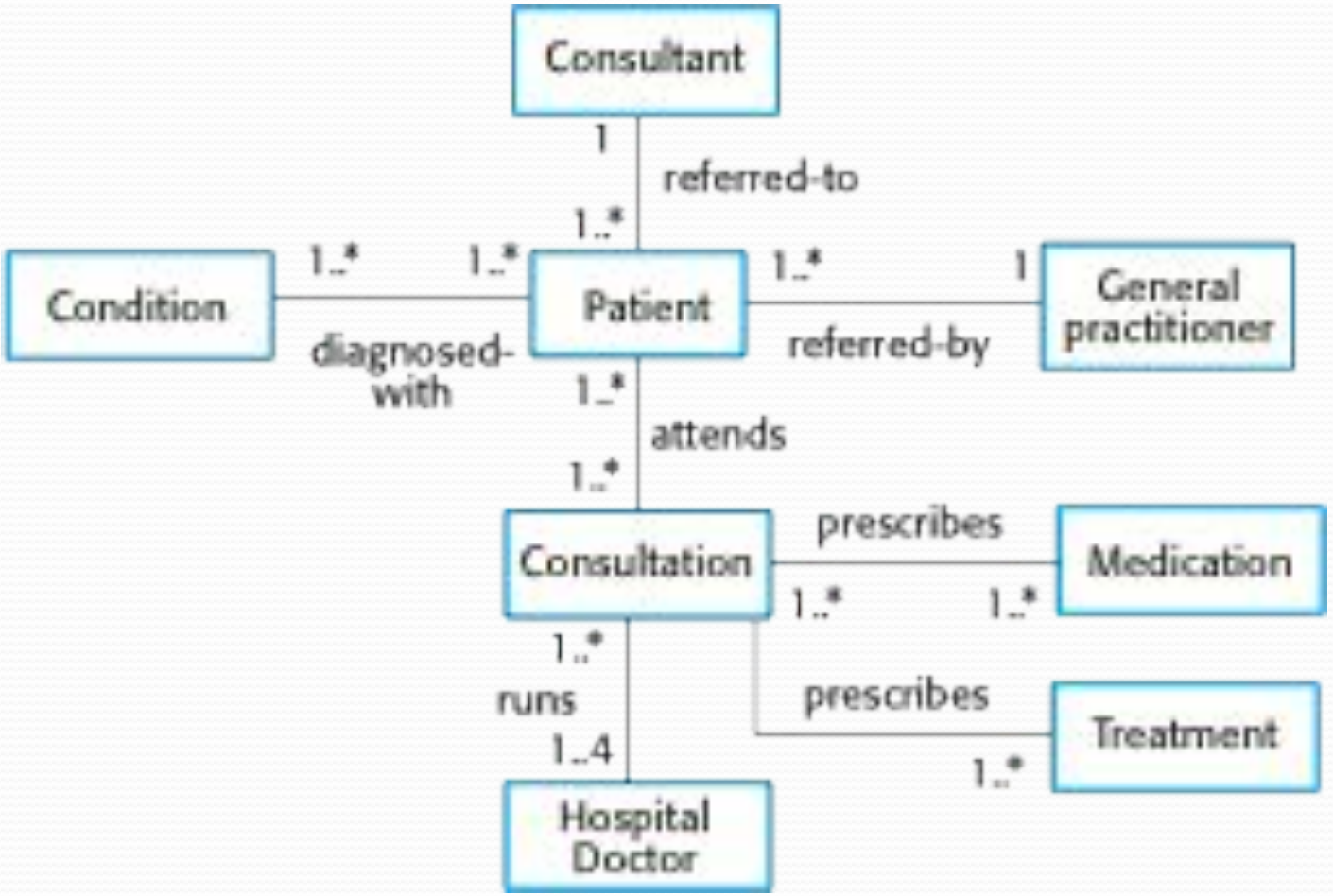
- Useful in designing the **system architecture**

Class diagrams

- Object-oriented system model
- Classes in a system
- Relationships between classes

UML classes and association

Classes and associations in the MHC-PMS

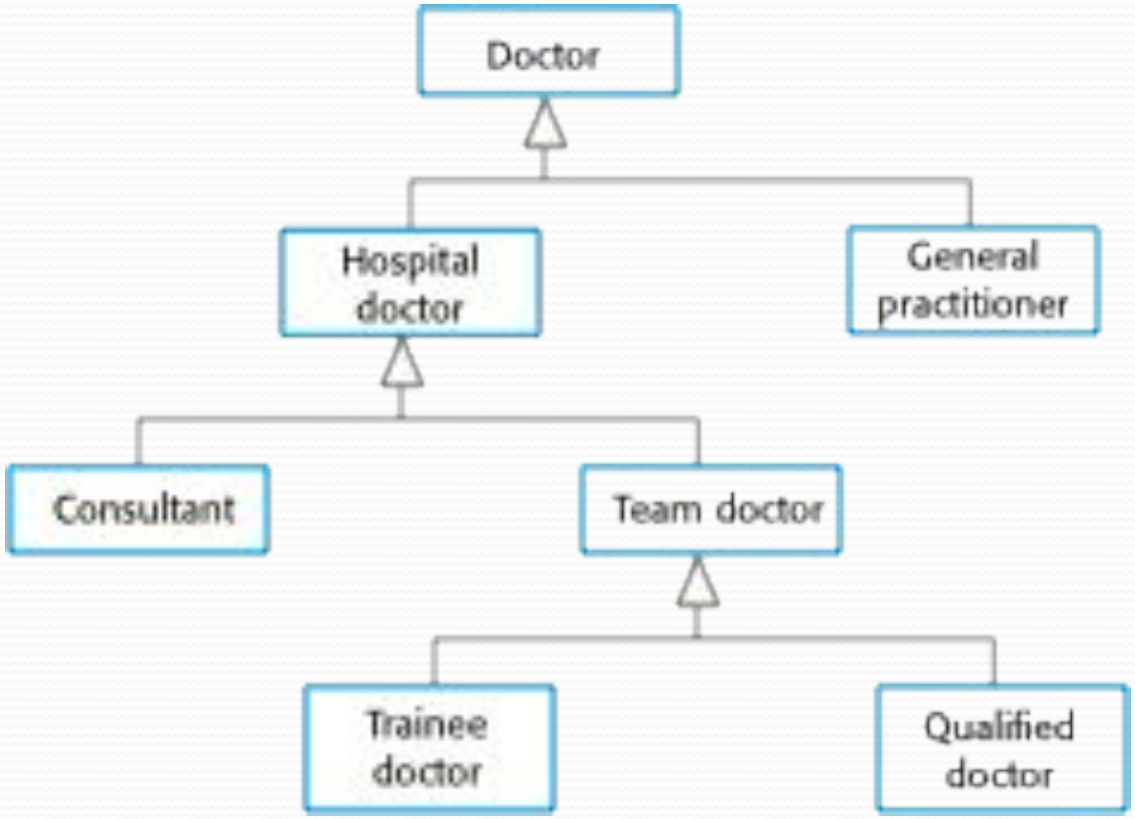


The Consultation class

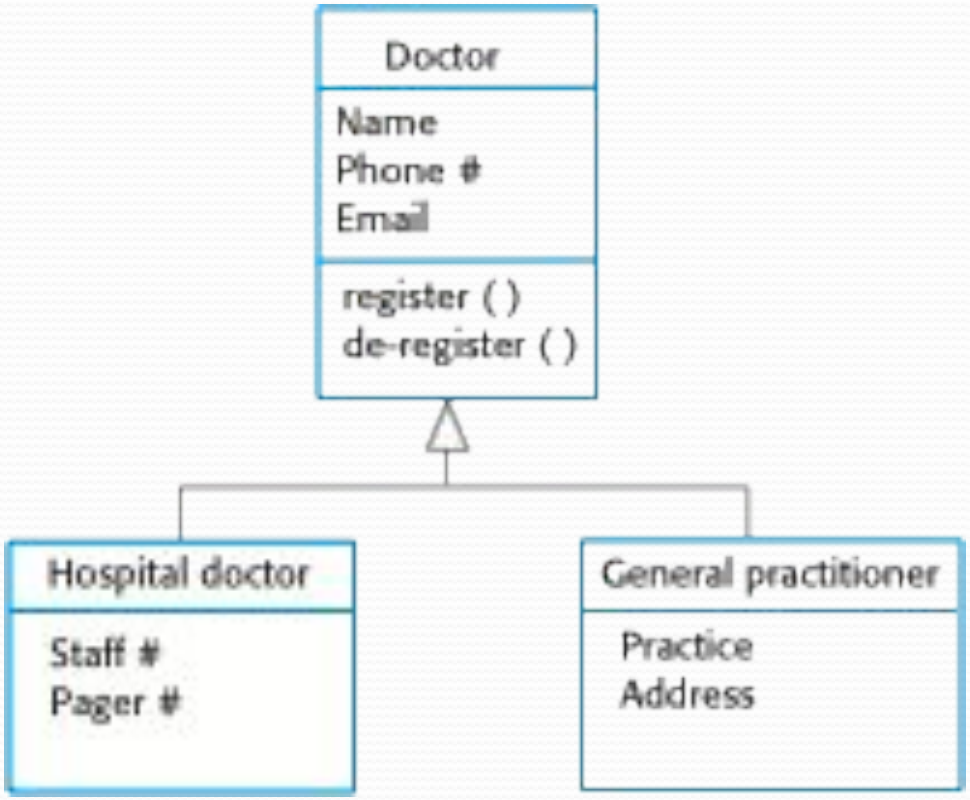
Generalization

- Represent shared properties and behavior
- OOP
  - Inheritance
  - Superclasses – shared properties
  - Low level classes have specific details

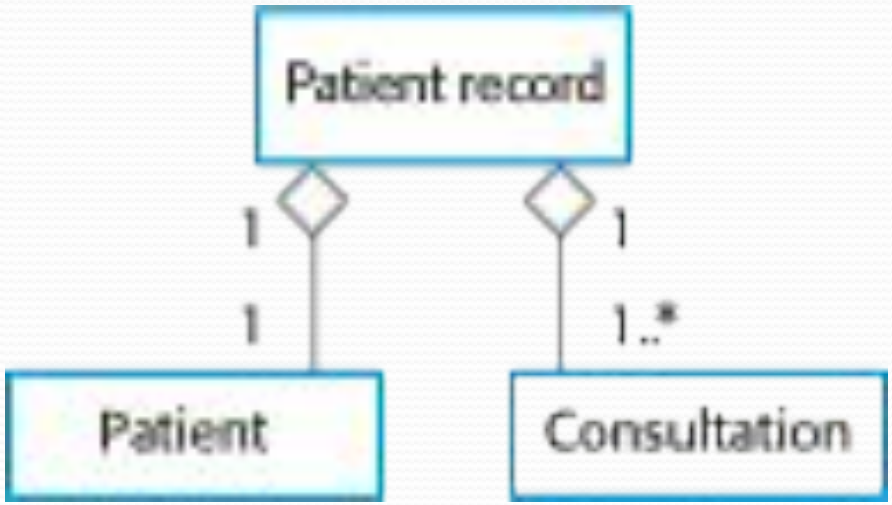
A generalization hierarchy



A generalization hierarchy with added detail

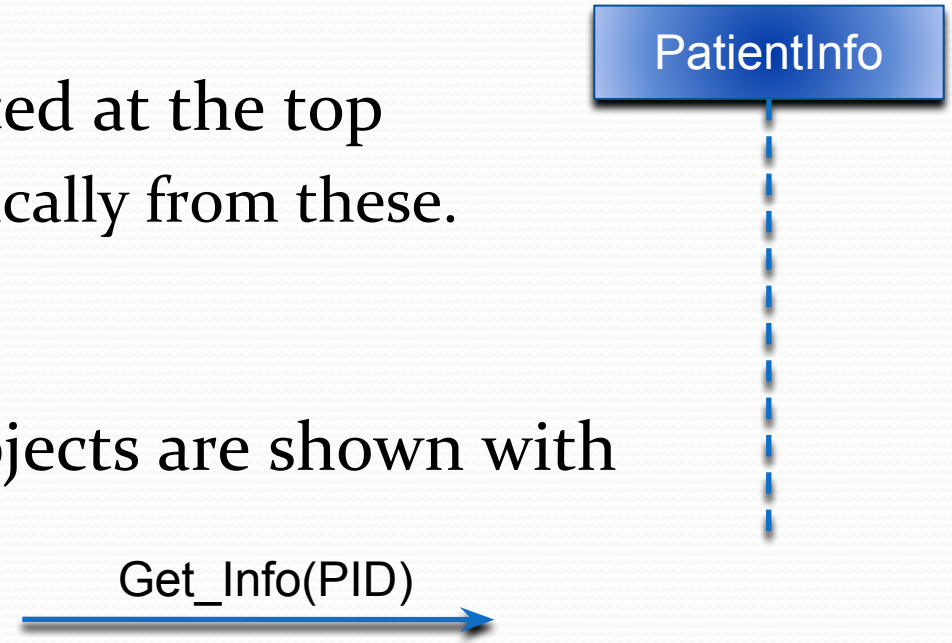


The aggregation association

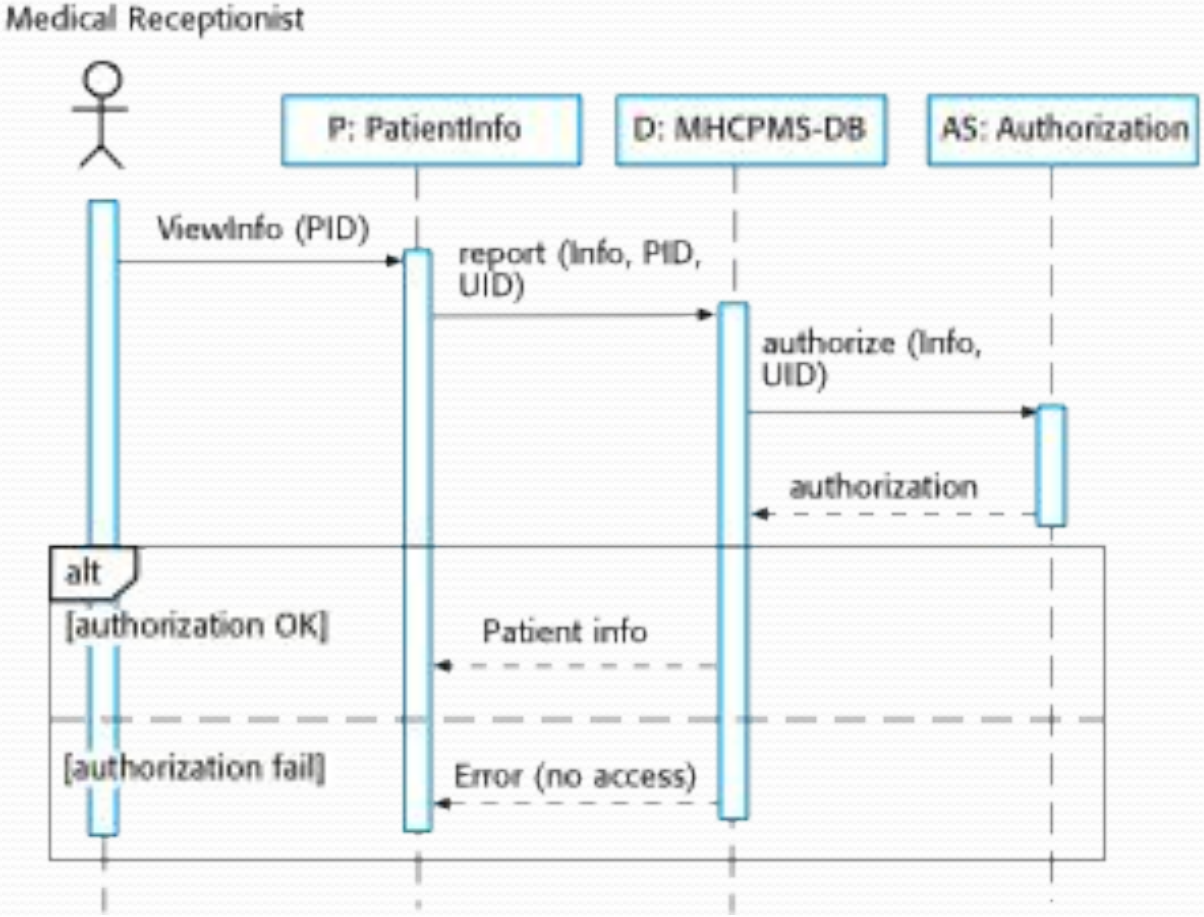


Sequence diagrams

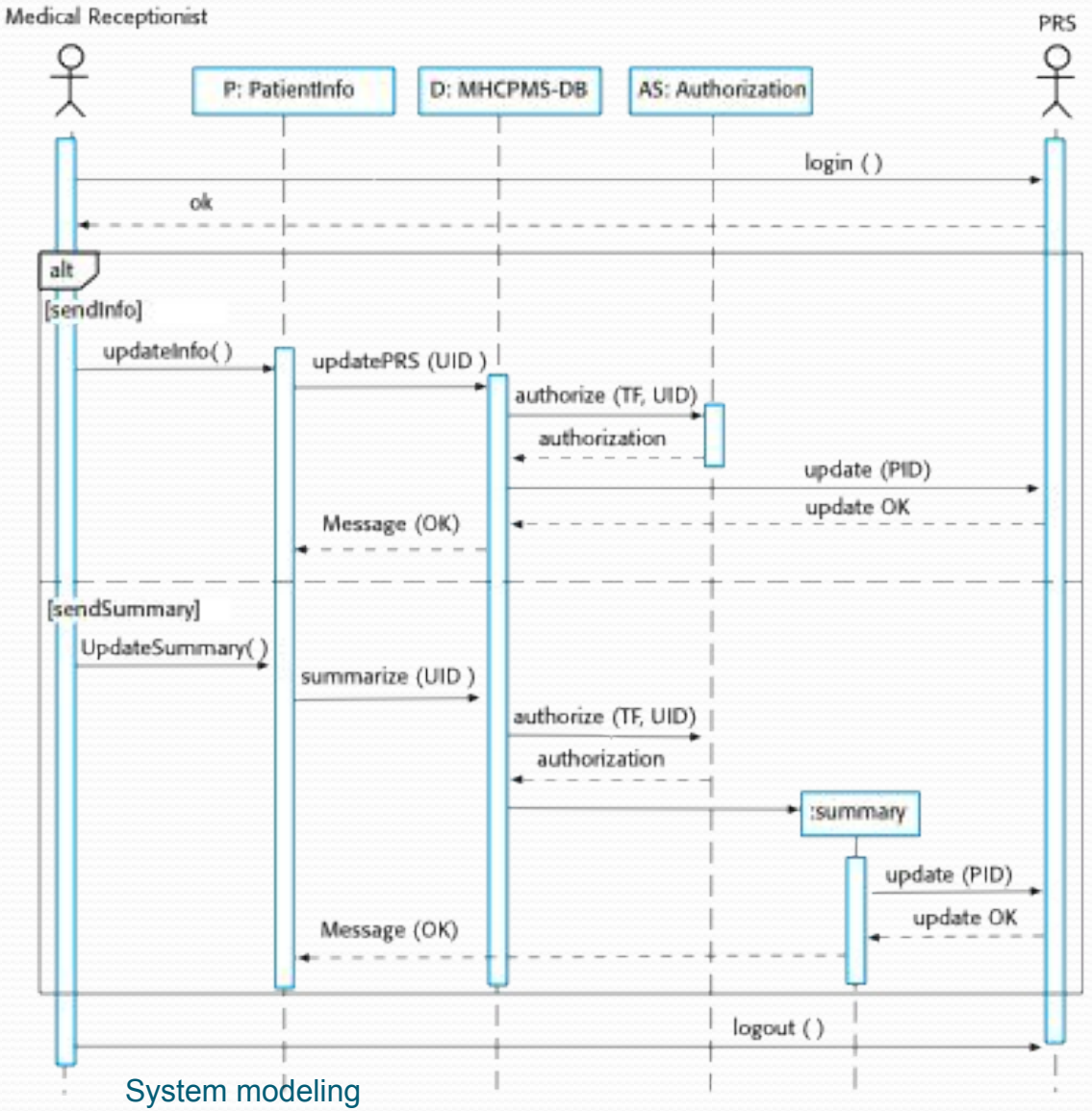
- **Interactions** between the **actors** and the objects within a **system**.
- Shows the sequence of interactions during a use case
- Actors & Objects are listed at the top
  - Dotted line drawn vertically from these.
- Interactions between objects are shown with annotated arrows



Sequence diagram for View patient information



Sequence diagram for Transfer Data



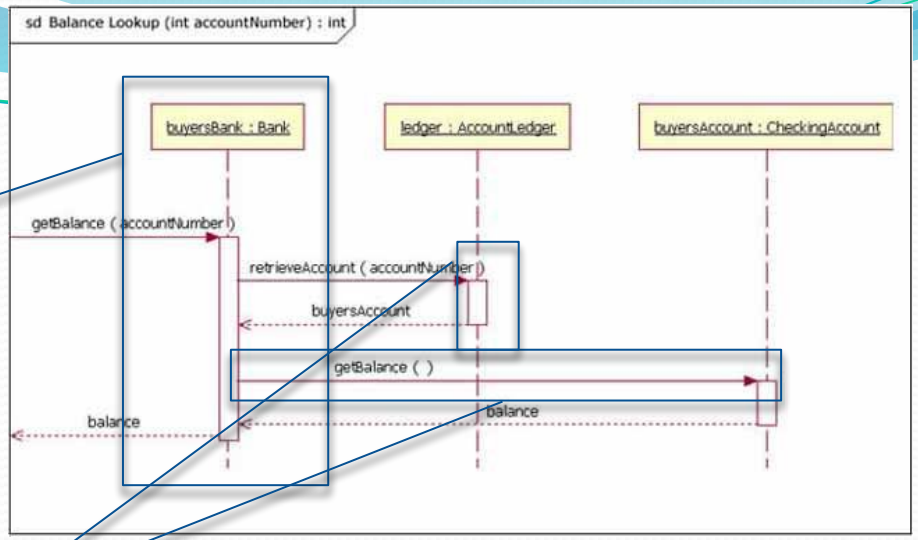
Interaction Modelling Notation

- Interaction **sequence** diagrams
- **Communication** diagrams
- Interaction **overview** diagrams
- **Timing** diagrams

Sequence Diagrams

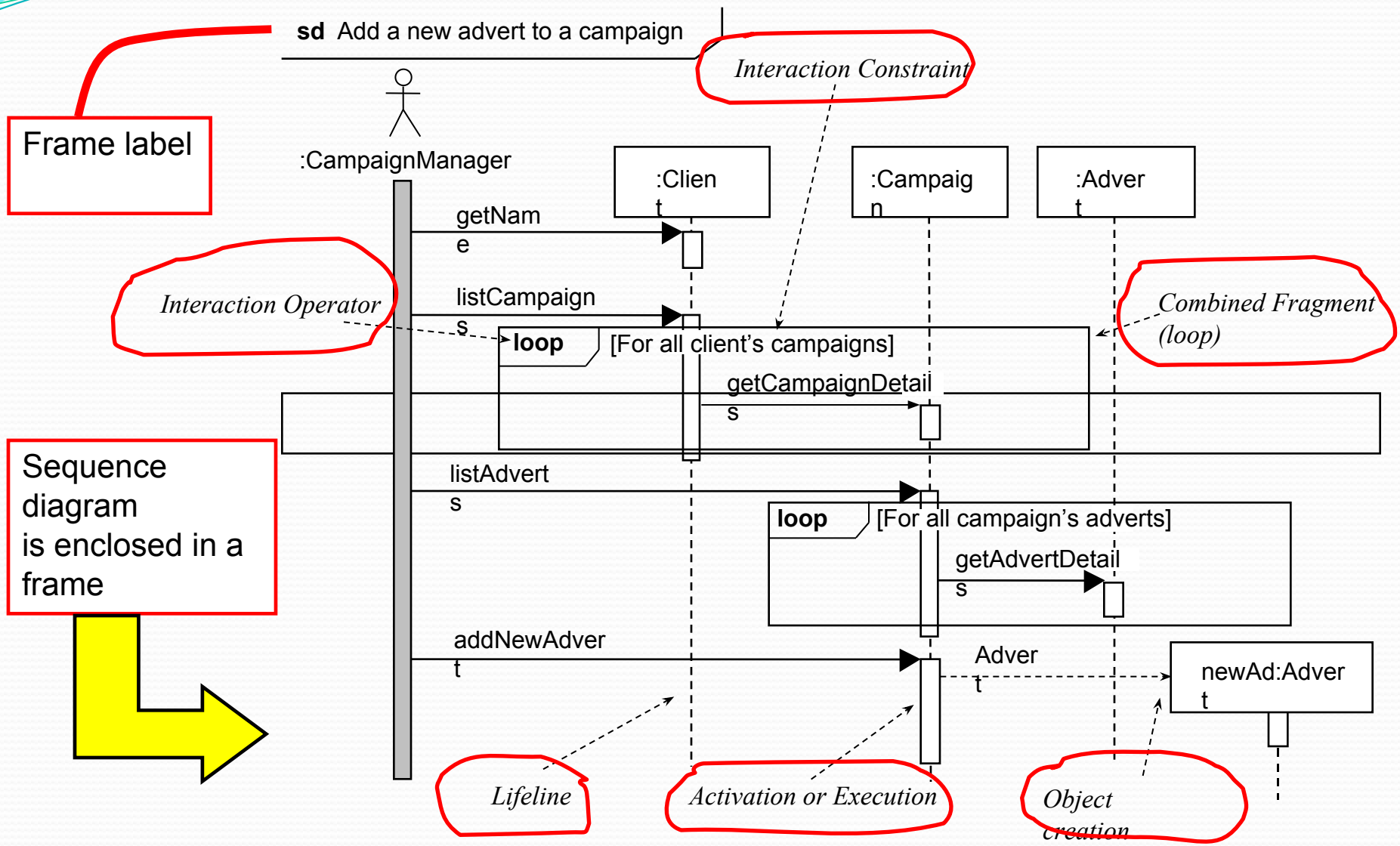
- Shows an interaction between objects with their lifelines
- Objects are arranged in a **time sequence**
- Typically shows object interaction for one use case

Sequence Diagrams



- **Lifelines**
  - Objects
  - Subsystems
- Lifelines are arranged horizontally.
- Time represented vertically
- **Messages** are represented with a solid horizontal arrow
- The **execution** or activation of an operation
  - represented with a rectangle
  - placed on relevant lifeline.

Sequence diagram

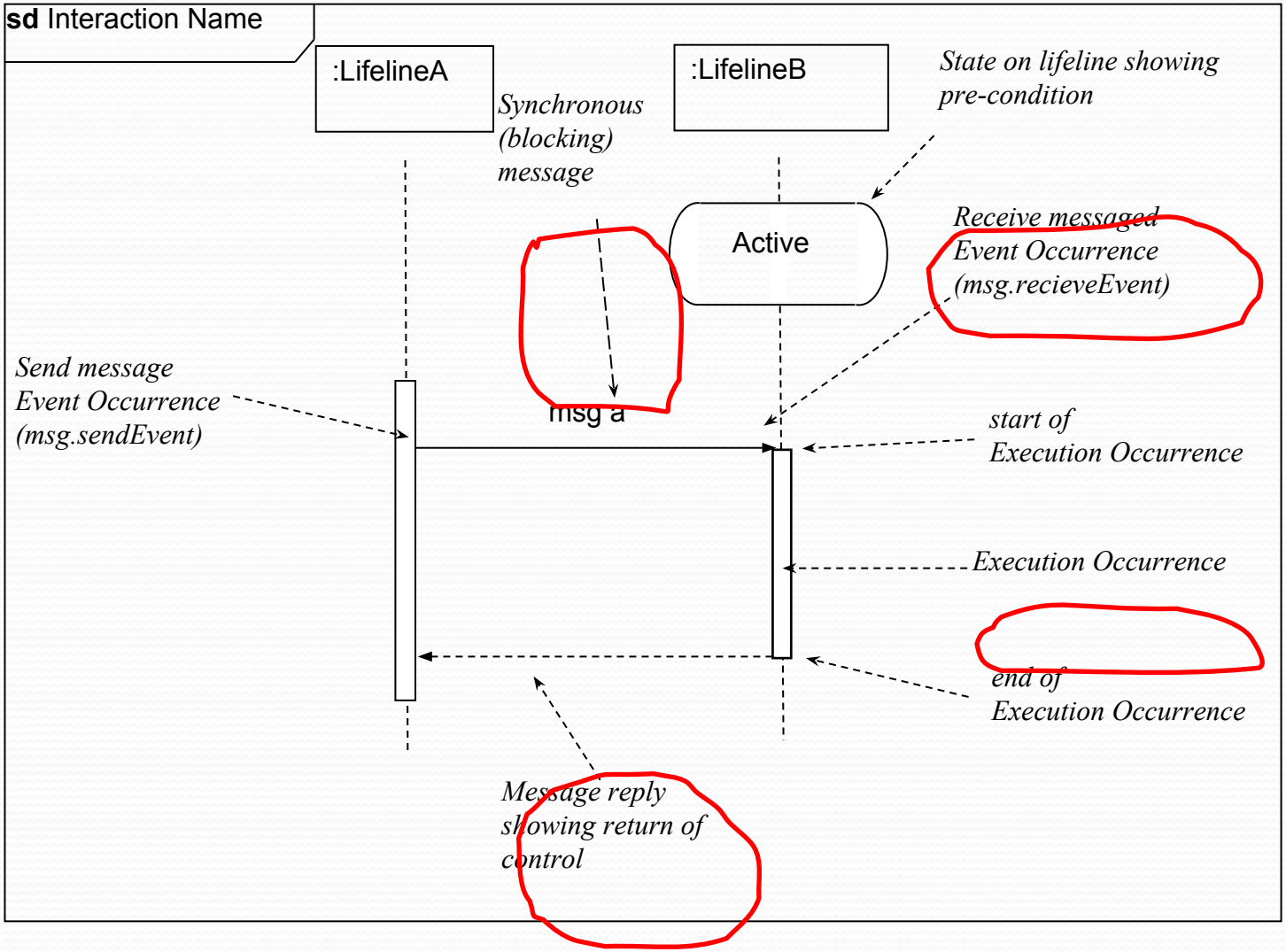


Sequence Diagram

- **Iteration** is represented with
  - *combined fragment* rectangle
  - *interaction operator* 'loop'
- The loop executes when guard condition evaluates to true
- Object creation
  - construction arrow (dashed)
  - Target: object symbol for the to lifeline

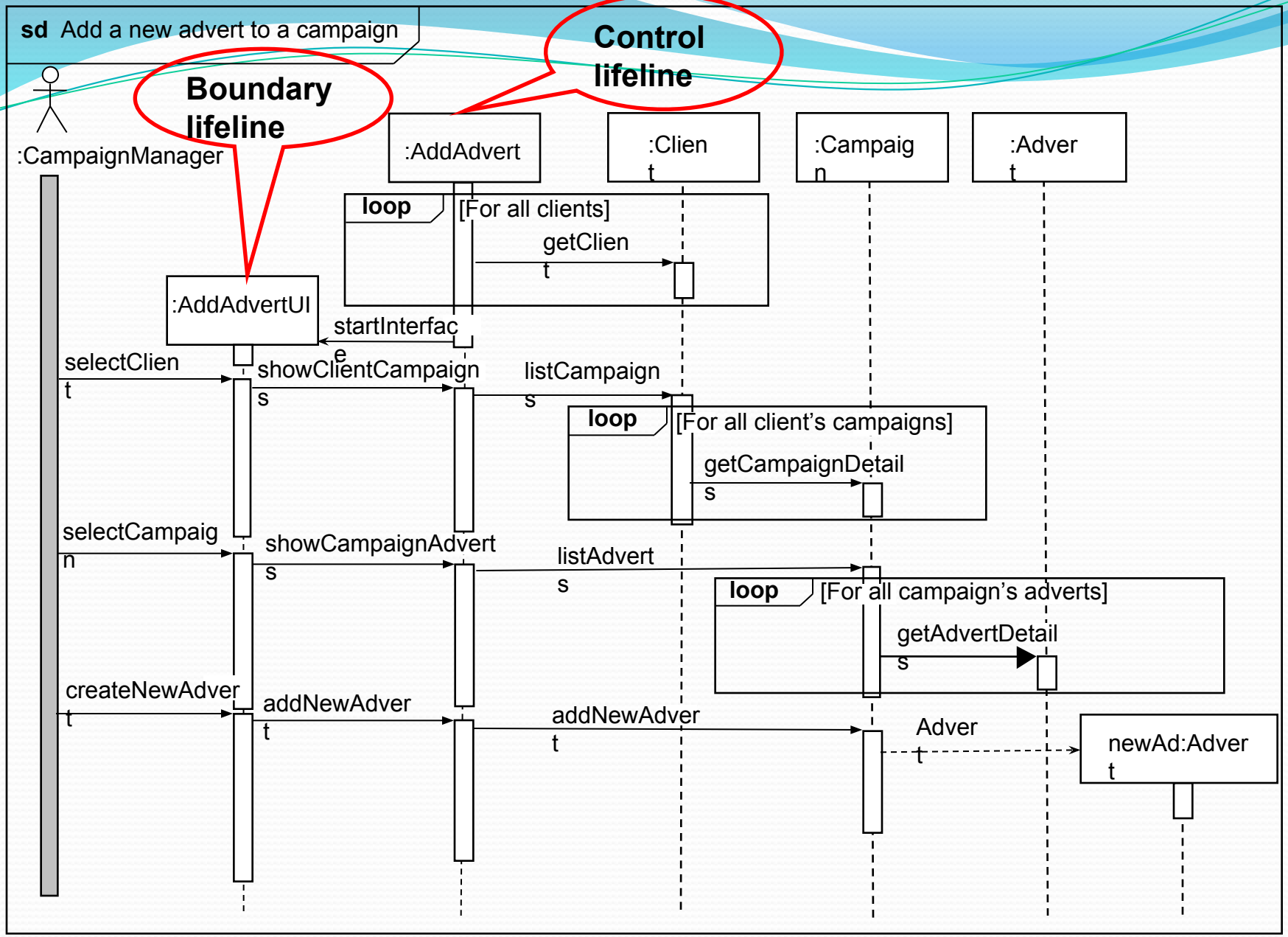
Synchronous message/procedural call

- Represented with a full arrowhead
- Caller execution is **suspended** until flow of control is returned to it.

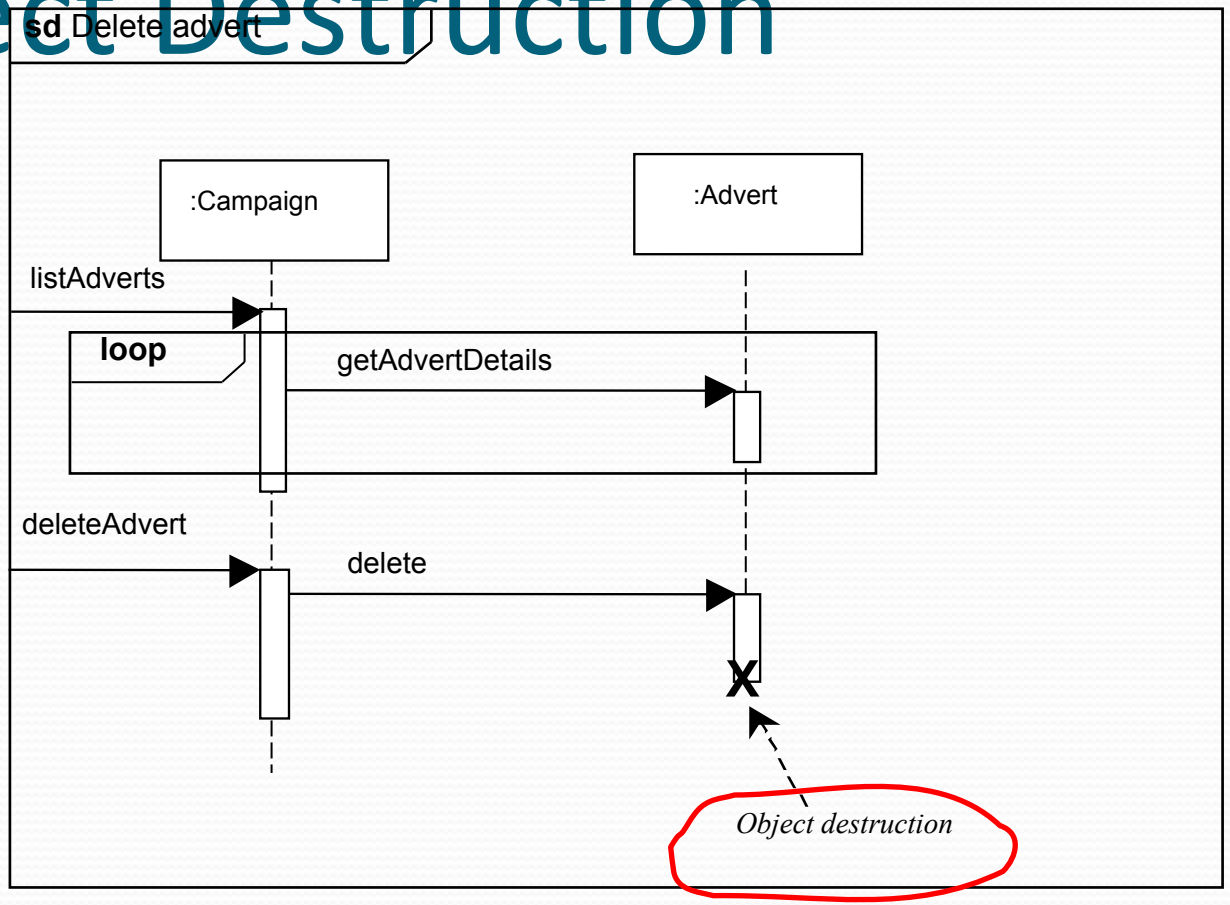

sd Interaction Name
:LifelineA
:LifelineB
Synchronous (blocking) message
State on lifeline showing pre-condition
Active
Receive messaged Event Occurrence (msg.recieveEvent)
msg a
Send message Event Occurrence (msg.sendEvent)
start of Execution Occurrence
Execution Occurrence
end of Execution Occurrence
Message reply showing return of control

Boundary & Control Classes

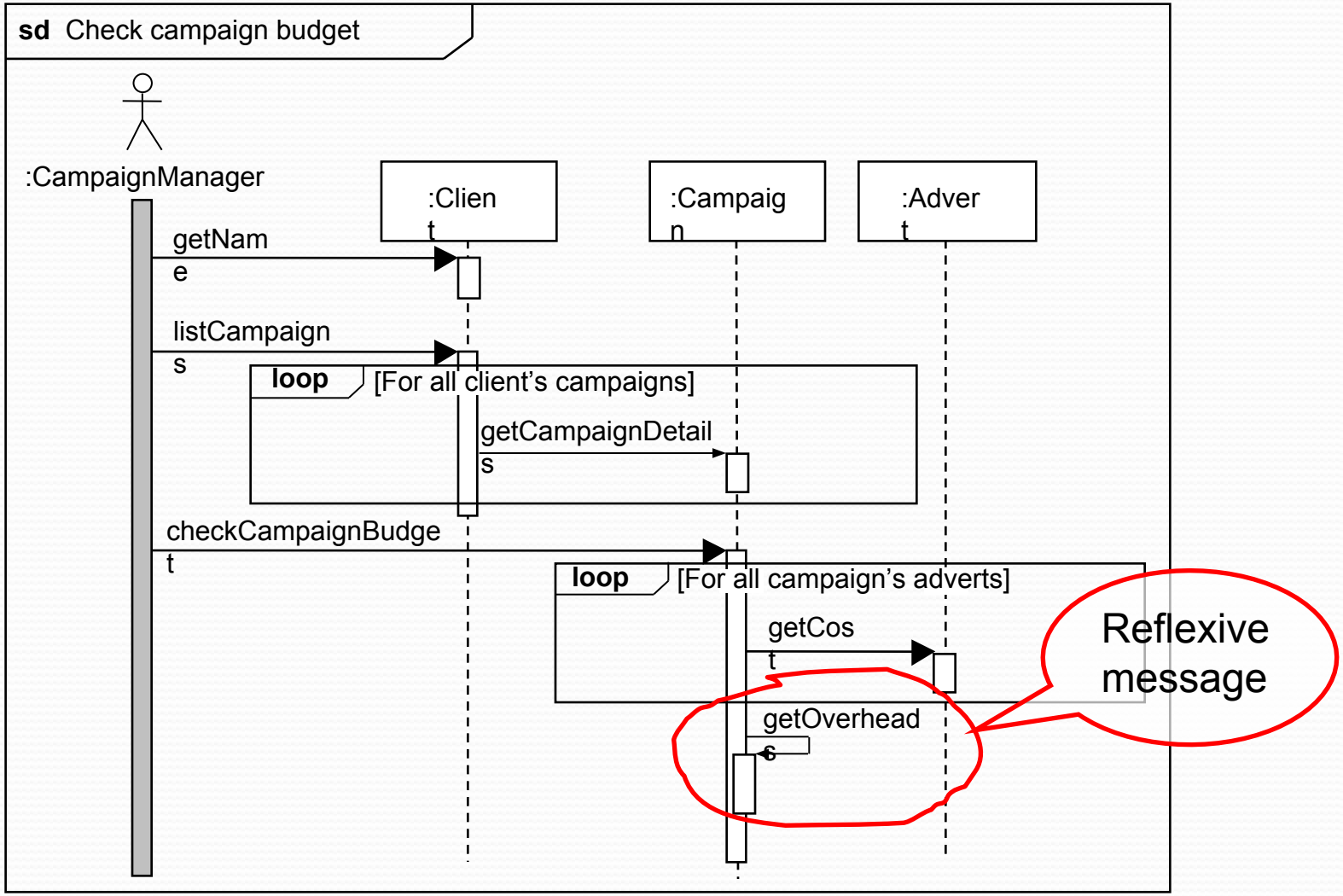
- Boundary object
  - manages interaction between the actor and the system
- Control manages overall object communication


sd Add a new advert to a campaign
Boundary lifeline
Control lifeline
:CampaignManager
:AddAdvert
:Clien t
:Campaig n
:Adver t
loop
[For all clients]
getClien t
:AddAdvertUI
startInterfac e
selectClien t
showClientCampaign s
listCampaign s
loop
[For all client's campaigns]
getCampaignDetail s
selectCampaig n
showCampaignAdvert s
listAdvert s
loop
[For all campaign's adverts]
getAdvertDetail s
createNewAdver t
addNewAdver t
addNewAdver t
Adver t
newAd:Adver t

Object Destruction



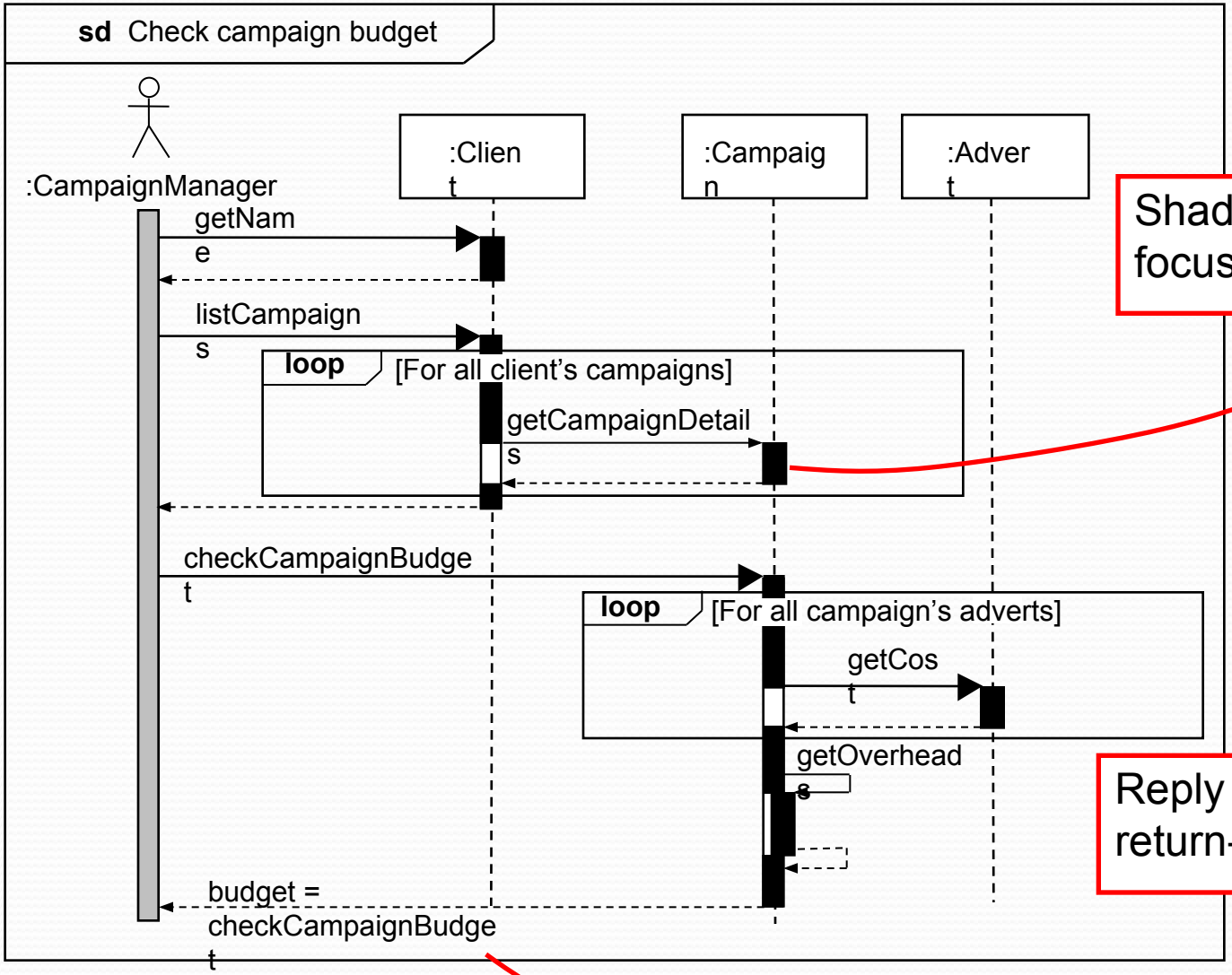
Reflexive Messages



Focus of Control

- Indicates when processing happens in an object (activation)
- When an activation is waiting, it does not have focus of control.

Focus of Control



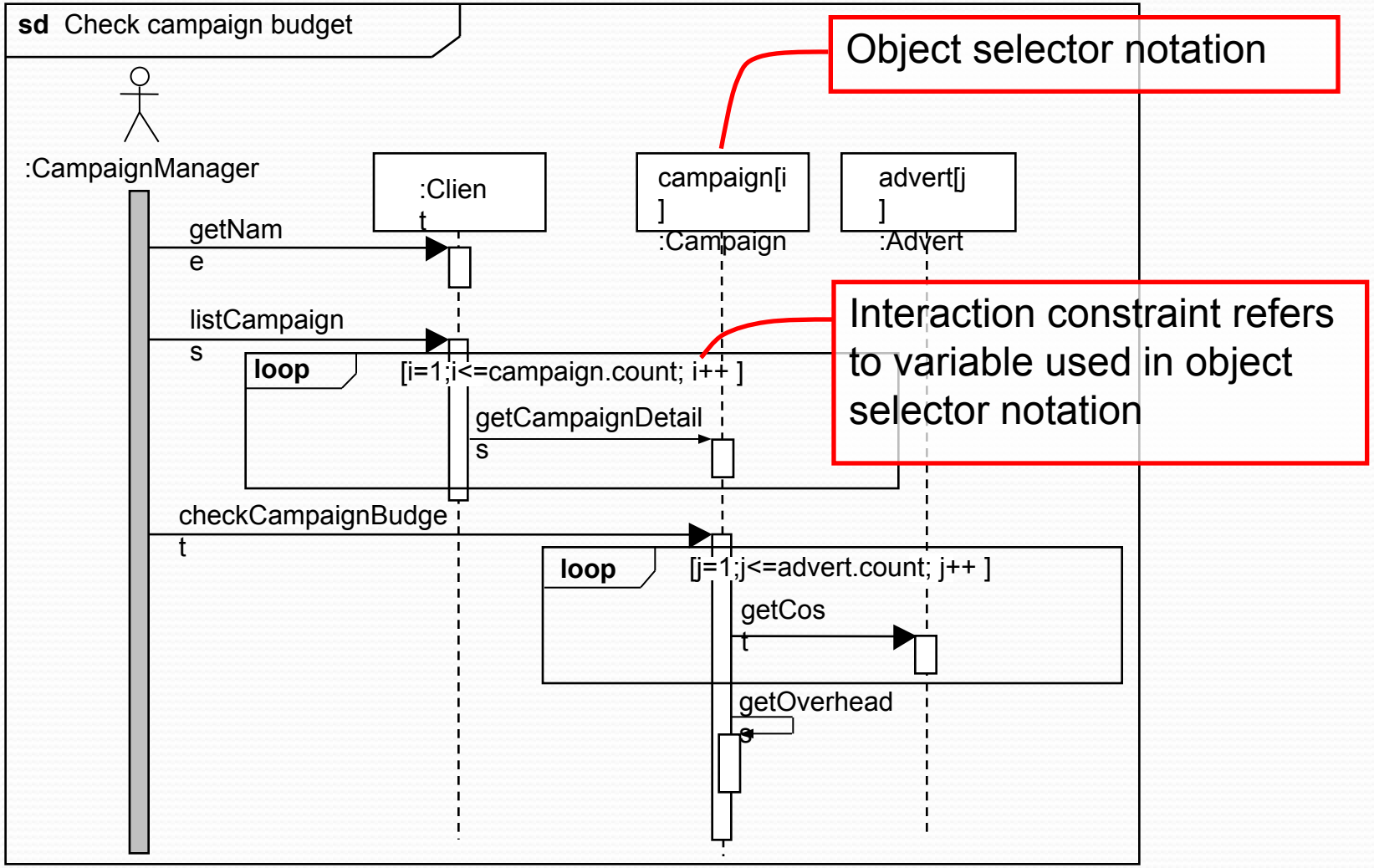
Shading showing the focus of control

Reply with the return-value shown

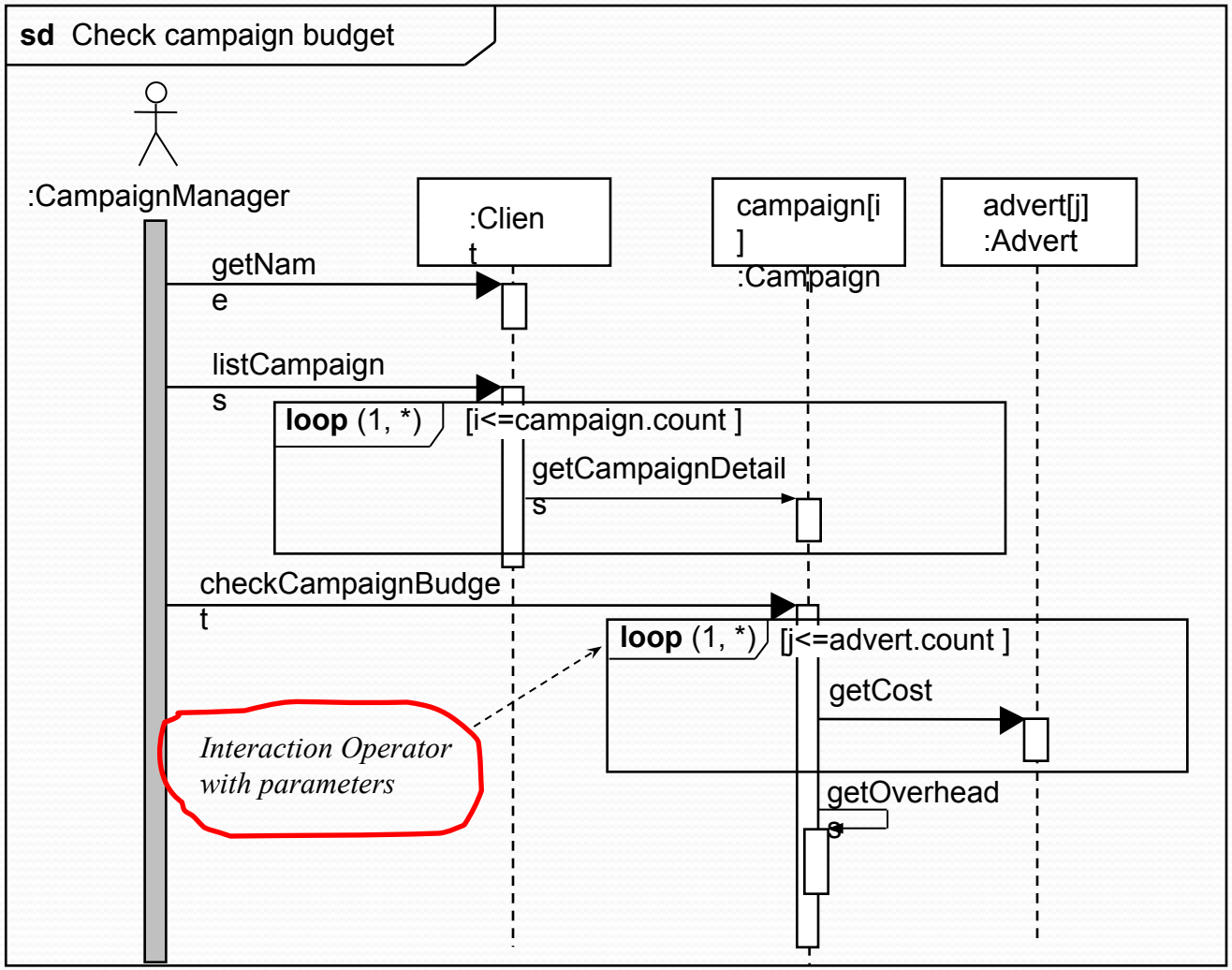
Reply Message

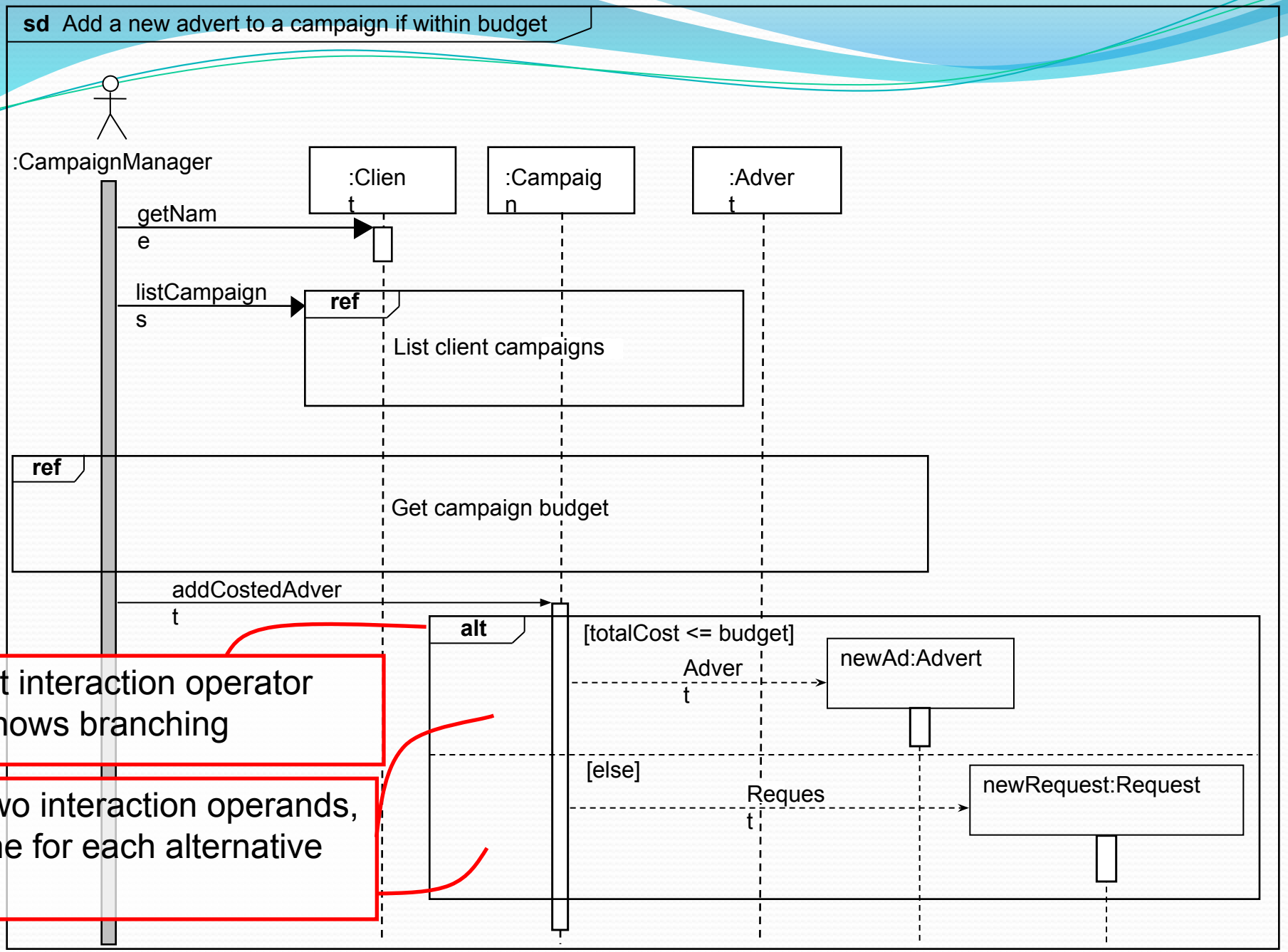
- A **reply message** returns the control to the caller object
- Shown with a dashed arrow
- Their use is optional
- Assume that control is returned to caller object

Object Selector Notation



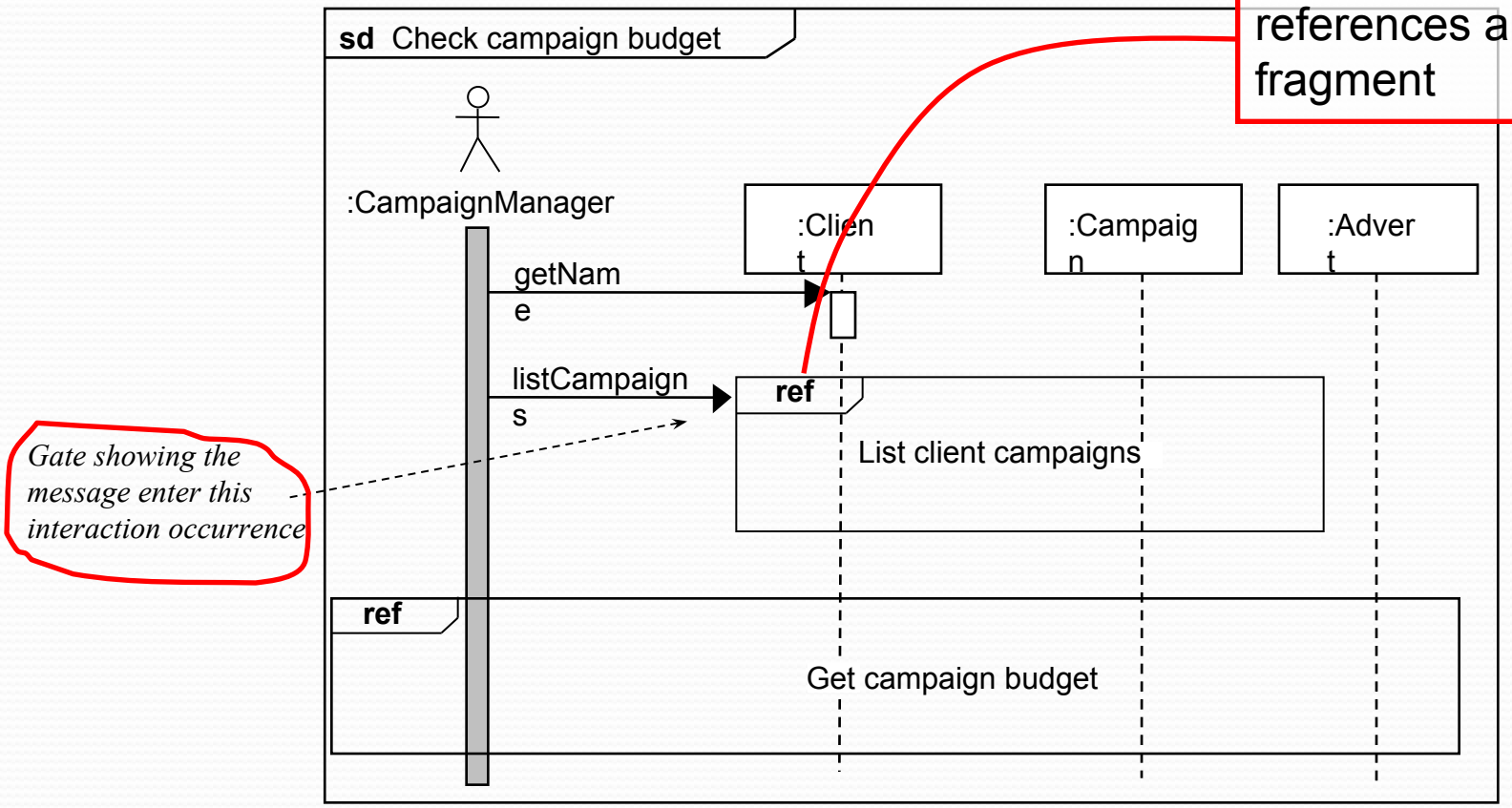
Interaction Operators




sd Add a new advert to a campaign if within budget
:CampaignManager
:Clien t
:Campaig n
:Adver t
getNam e
listCampaign s
ref
List client campaigns
ref
Get campaign budget
addCostedAdver t
alt
[totalCost <= budget]
Adver t
newAd:Advert
[else]
Reques t
newRequest:Request
alt interaction operator shows branching
Two interaction operands, one for each alternative

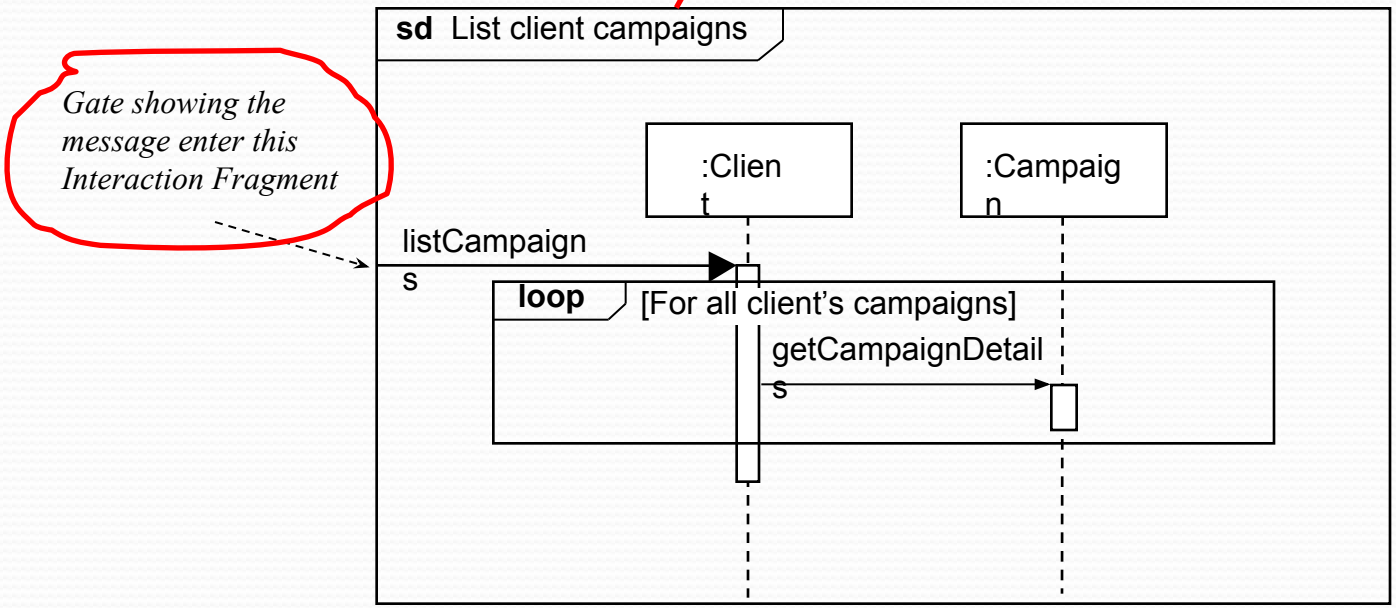
Using Interaction Fragments

ref interaction operator indicates interaction occurrence that references an interaction fragment



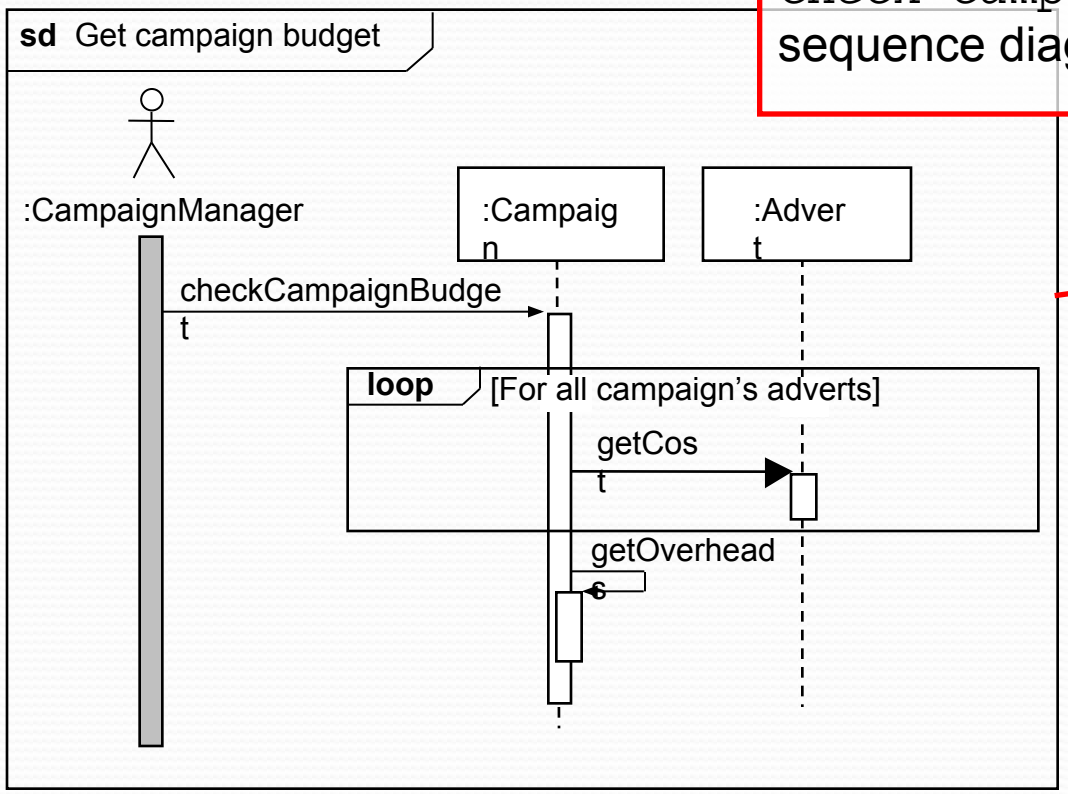
Interaction Fragment

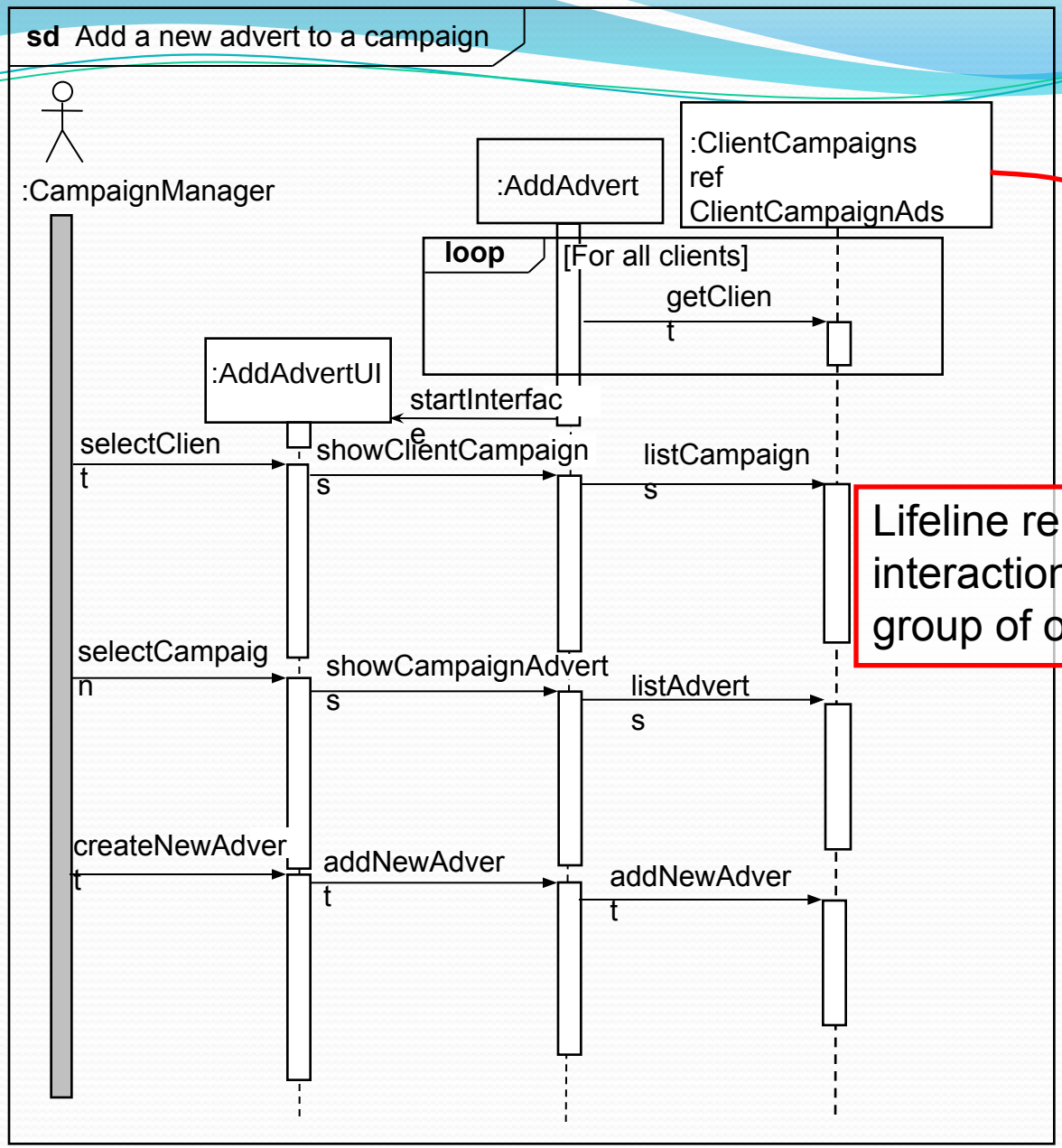
Interaction fragment that is referenced in `Check campaign budget` sequence diagram



Interaction Fragment

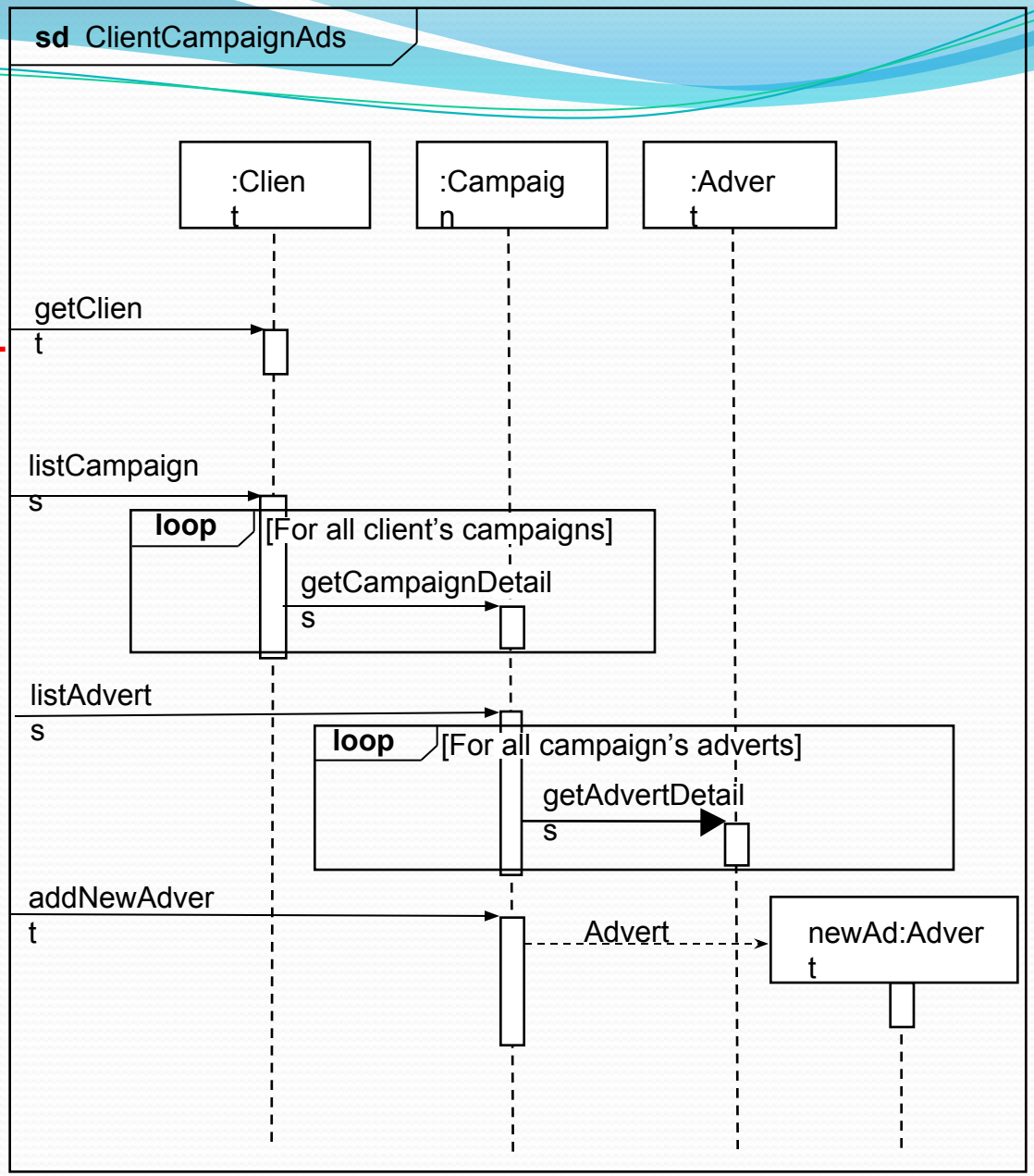
Interaction fragment that is also referenced in `Check campaign budget` sequence diagram



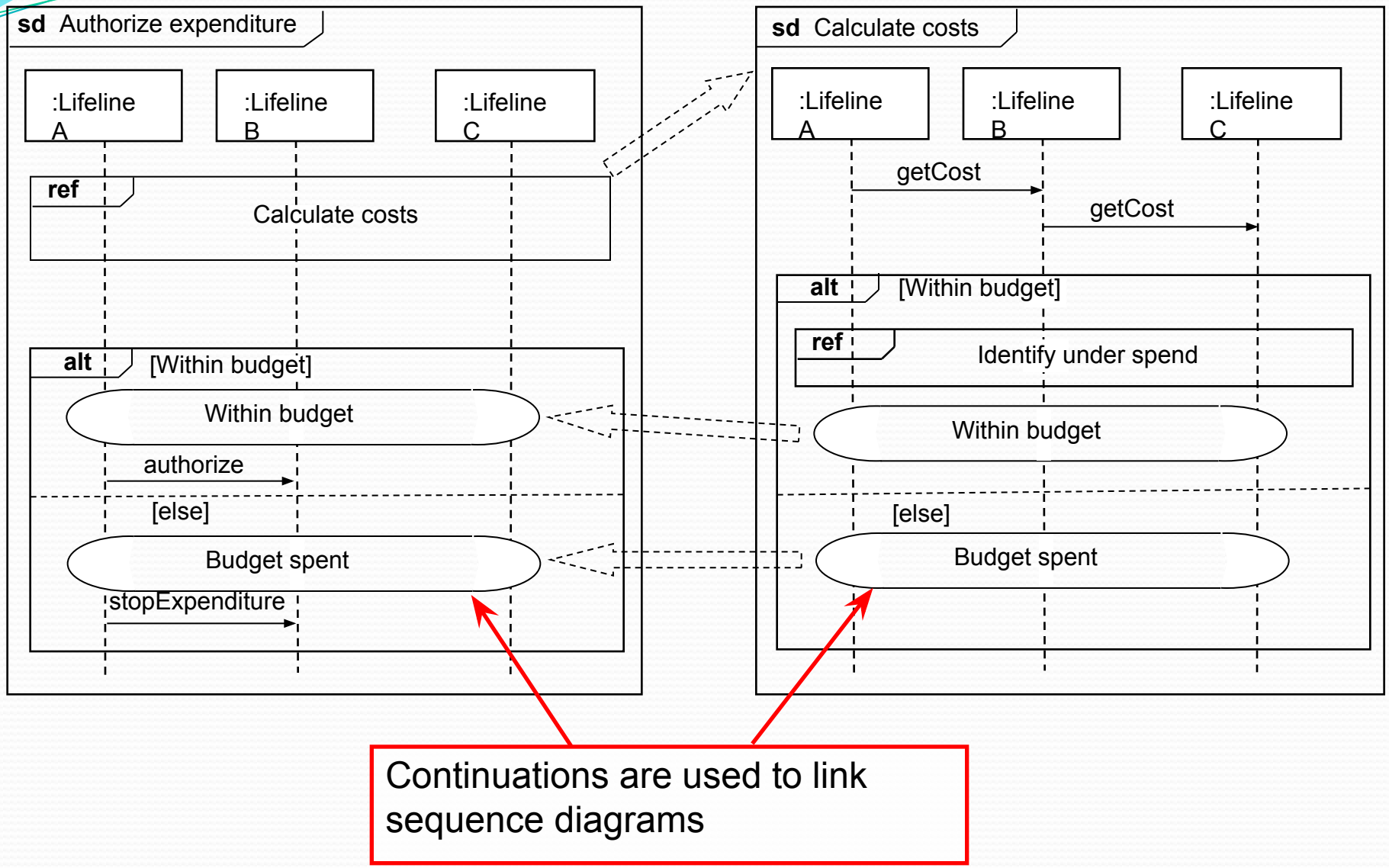


Lifeline representing the interaction between a group of objects

Sequence diagram referenced in the `Add a new advert to a campaign` sequence diagram



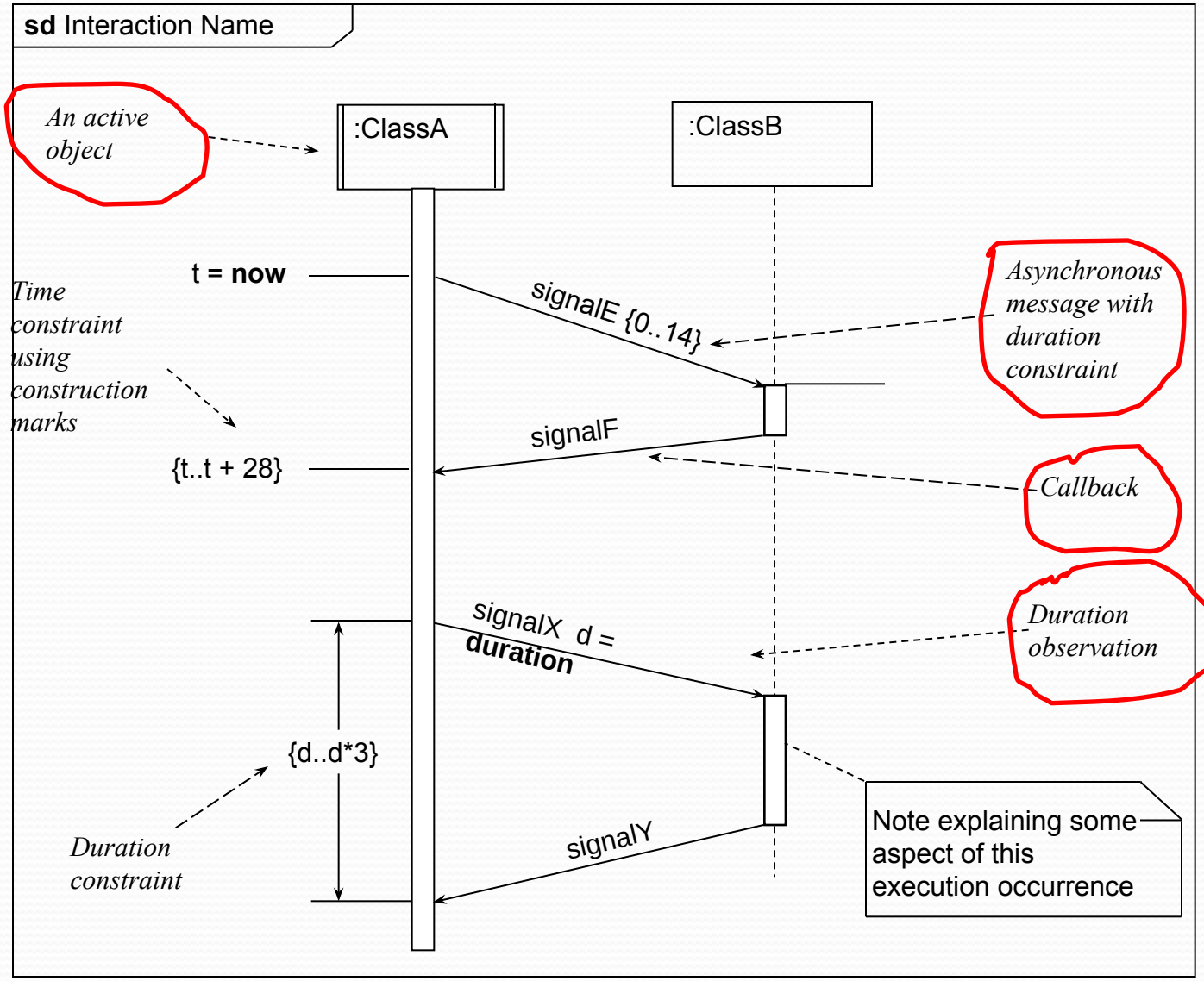
Using Continuations



Asynchronous Message

- *Asynchronous message*
  - drawn with an **open arrowhead**
  - does not halt execution to wait for a return
- Real-time systems
- Concurrency
- Callback to caller can be used to signal completion

Time – by giving time or duration



Interaction Operators

Interaction Operator	Explanation and use
alt	Alternatives represents alternative behaviours, each choice of behaviour being shown in a separate operand. The operand whose interaction constraint is evaluated as true executes.
opt	Option describes a single choice of operand that will only execute if its interaction constraint evaluates as true.
break	Break indicates that the combined fragment is performed instead of the remainder of the enclosing interaction fragment.
par	Parallel indicates that the execution operands in the combined fragment may be merged in any sequence once the event sequence in each operand is preserved.
seq	Weak Sequencing results in the ordering of each operand being maintained but event occurrence from different operands on different lifelines may occur in any order. The order of event occurrences on common operands is the same as the order of the operands.
strict	Strict Sequencing imposes a strict sequence on execution of the operands but does not apply to nested fragments.
neg	Negative describes an operand that is invalid.
critical	Critical Region imposes a constraint on the operand that none of its event occurrences on the lifelines in the region can be interleaved.
ignore	Ignore indicates the message types, specified as parameters, that should be ignored in the interaction.
consider	Consider states which messages should be consider in the interaction. This is equivalent to stating that all others should be ignored.
assert	Assertion states that the sequence of messaging in the operand is the only valid continuation.
loop	Loop is used to indicate an operand that is repeated a number times until the interaction constraint for the loop is no longer true.

Guidelines for Sequence Diagrams

1. Decide a level of modelling the interaction
2. Identify the main elements involved in the interaction
3. Identify alternative scenarios
4. Draw an outline
5. Refine with detail