

## UML Diagram Types

<b>Dynamic Models</b>	<b>Structural Models</b>
■ activity diagrams	■ class diagrams
■ statechart diagrams	■ object diagrams
■ <i>interaction diagrams</i>	■ packages
– <i>sequence diagrams</i>	<b>Architectural Models</b>
– <i>collaboration diagrams</i>	■ component diagrams
■ use case diagrams	■ deployment diagrams

---

---

---


---

---

---

---

---



## Interaction

*def'n:* behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose

- represents the dynamic behavior of *objects*
- can model flow of control within operation, class, component, use case, or system
- can model simple sequential flow, branching, looping, recursion, concurrency
- time order: sequence diagram
- structural order: collaboration diagram

---

---

---


---

---

---

---

---



## Context of Interaction Diagram

- System: higher level of abstraction
- Operation: parameters, local variables, global objects
- Class: semantics of class
- Use Case: diagrammatic representation of scenario

---

---

---

---

---

---

---

---

## Message

*def'n*: specification of a communication between objects that conveys information with the expectation that activity will ensue

- receipt of message is occurrence of event

Convention

- create      <<create>> →
- destroy     <<destroy>> → ✕
- call         opname() →
- return      value →
- signal      signal →

---

---

---

---

---

---

---

---

## Call and Return

*Call def'n*: invoke an operation on an object

- object may send message to self (local invocation)
- object *pointed to* owns operation

*Return def'n*: return a value to caller

Convention

- opname() →
- value →

---

---

---

---

---

---

---

---

## Create and Destroy

*Create def'n*: create an object

*Destroy def'n*: destroy an object (an object may destroy itself)

Convention

- <<create>> →
- <<destroy>> → ✕

---

---

---


---

---

---

---

---



### Signal

*def'n*: named object that is dispatched asynchronously by one object and received by another

Convention

→  
signal

---

---

---


---

---

---

---

---



### Message Naming

Typical Form

- sequence number:operation(argument)

Examples

- 2: clickAt(p)
- 2.1: I = findAt(p)
  - I is implied returned value
- D5: ejectHatch(3)
  - fifth message of root process D calling operation *ejectHatch* with argument 3

---

---

---


---

---

---

---

---



### Interaction Diagram

*def'n*: interaction, consisting of a set of objects and their relationships, including messages dispatched among them

- comprised of object, links, and messages
- context is a scenario that illustrates behavior
- may model one particular flow of control of use case

---

---

---


---

---

---

---

---



## Sequence Diagram

*def'n:* interaction diagram that emphasizes time ordering of messages

- has close relationship to use case diagram
- x axis
  - objects
  - object that initiates to left
  - increasingly more subordinate to right
- y axis
  - messages ordered in increasing time from top to bottom

---

---

---


---

---

---

---

---



## Collaboration Diagram

*def'n:* interaction diagram that emphasizes the structural organization of the objects that send and receive messages

- has close relationship to object diagram
- vertices: representing objects
- arcs: representing links and/or messages passing between objects

---

---

---


---

---

---

---

---



## Sequence vs. Collaboration Diagram

Semantically equivalent, except:

- Sequence diagrams have:
  - object lifelines: vertical dashed line representing life of object
  - focus of control: tall, thin rectangle showing period of time which an object is performing an action
- Collaboration diagrams have:
  - path: indicates that one object is linked to another
  - sequence number: indicate time ordering of message

---

---

---

---

---

---

---

---



### To Model Sequence Diagrams

- Determine context
- Identify objects and layout from L to R
- Set lifeline of each object
- Lay out messages within lifelines
- Adorn with focus of control

---

---

---

---

---

---

---



### To Model Collaboration Diagrams

- Determine context
- Draw object diagram
- Set initial properties of each object
- Specify links between objects
- Add messages to appropriate links
- Add sequence numbering

---

---

---

---

---

---

---