Interaction Diagrams

- Examples of Collaboration and Sequence Diagrams

Dynamic Views in UML

- Class diagrams are models of data types
  - What non-fundamental types are you using? How are they related?
  - Sometimes referred to as the static model
- Running programs define objects of various types that interact
  - Control is passed between objects’ methods
  - Information is passed and returned
- UML has two almost identical diagrams for this:
  - Collaboration diagram: object-centered
  - Sequence diagram: time-oriented
- Important: Each diagram models a particular scenario
  - Often we just model important or interesting scenarios
Class Diagram with a Related Collaboration

Sequence Diagram: Ex. 1
Use Cases and Sequence Diagrams

- So far sequence diagrams model interactions between implementation-level objects
- Also, can be used to model use cases
  - Actor(s) interacting with the System
    - Most useful if more than one Actor
    - Messages are used informally here
  - Example: ATM customer tries to overdraw
- Also, sequence diagrams sometimes used in high-level design
  - model interactions between major subsystems
Iteration, Control, Return Values

• Messages can be labeled with a condition:
  [hasStock] sellWidget()
• Messages can be repeated:
  * msg() or *[k=1..2] msg()
• Return values:
  – Maybe on dashed “return” arrow, or
  – On message call: n = getName()
  – Again, note returns not always explicitly drawn

Notes on Messages

• Various types of messages supported
  – Filled solid arrow head:
    • like procedure calls, nested flow of control.
      Caller waits for action to complete.
  – “Half” arrow head:
    • Asynchronous flow of control. Caller is not
      blocked, continues to do something.
  – Dashed arrow:
    • Return from procedure call. May be omitted
      if it’s implicit at end of an activation box
    • Use stereotypes to define anything else.
Iterations, Constraints, Asynch. Messages

Phone Switch Example

- Optional for class lecture...
Phone Switch Example

• Classes
  – Phone: a person’s telephone
  – Line: a "number" associated with a phone
    • Lines are busy, calls are made on lines, etc.
  – Connection: dynamically created, represents an active call between lines
  – Switch: a phone switch is a computer system that manages phones, lines and connections
  – Console: a terminal attached to the switch
  – Simulator: we’re writing a simulation!

Associations in Our Model

• Switch manages other objects
• More than one Phone may be “linked” to one line
  – Like for office secretary, boss
• A Phone may have more than one Line
  – But, a Phone has only one Line “selected” at one time
    • Must choose a Line to call or answer
  – Also, when a Line is connected, not all Phones that can possibly use that Line may be participating in the call.
• A Connection requires at least two Lines to exist
Conceptual Class Diagram

Scenario

• A normal call
  – Phone1 chooses Line 1 and picks up
  – Phone 1 dials number for Phone2
  – Phone 2 rings
  – Phone 2 picks up
  – Call completed and the two people talk
  – Phone 2 hangs up
  – Phone 1 is disconnected
  – Phone 1 hangs up
Class Diagram with Operations

Simulator
- processCmds(in : istream)
  1

Switch
- name : string
- getName(name : string) : Phone
- getLine(number : string) : Line
- addLine(num : String) : void
- addPhone(p : Phone) : void
- newCall(l1 : Line, l2 : Line) : void
- completeConn(l : Line, c : Connection) : void
- destroyConn(c : Connection) : void

Connection
- id : int
- state : ConnState
- disconnect(l : Line) : void
- addLine(l : Line) : void

Phone
- state : PhoneState
- name : string
- selectLine(num : int) : void
- pickup() : void
- hangup() : void
- hangup(p : Phone) : void
- ring() : void
- dial(num : int) : void
- addLine(l : Line) : void

Line
- number : int
- state : LineState
- incomingCall(c : Connection) : void
- hungUp(p : Phone) : void
- pickedUp(p : Phone) : void
- addPhone(p : Phone) : void

Switch manages
- Line
- Phone

Simulator controls
- Switch
- Line
- Phone

Sequence Diagram for a Normal Phone Call

1. Simulator
2. Line
3. Switch
4. Line
5. Line
6. Phone
7. Line
8. Phone
9. Phone
10. Line
11. Line

selectLine(n)
pickup()
dial(num)
newCall(l1, l2)
pickedUp(p1)
incomingCall(c1)
hangup()
hungup(p1)
ring()
pickedUp(p2)
completeConn(l2,c1)
addLine(l2)
hangup()
hungUp(p2)
disconnect(l2)
destroyConn(c1)