Dynamic Modeling

R K Joshi
Dept. of Computer Science & Engineering
IIT Bombay
Building complex systems

• Model first

• Then build and deploy
Modeling

• Modeling with different perspectives
  – Static vs. dynamic; system to variables; requirements to deployment; ...

• No single view may be sufficient

• A Set of models for a complex system
• Nearly independent views
• But views must be consistent with each other
Modeling language

• Models need to be represented

• Modeling languages
  – Supports different models
  – Notations for models
  – Semantics need to be defined
  – Generation of code from models
  – Traceability into implementation

• UML is a modeling language
Dynamic modeling

• Capturing changes
  – States
  – Invocations
  – Flow of events

Aspects to be considered
  dependency
  concurrency
  participation
UML support for modeling

• Use case diagram
• Class diagram
• Statechart diagrams
• Activity diagram
• Sequence diagram
• Collaboration diagram
• Component diagram
• Deployment diagram
A Sequence diagram

- caller
  - a: pickup receiver
  - b: dialtone
  - c: dial digit
  - .......
  - Ringing tone
  - Stop tone

- exchange

- receiver
  - Phone rings
  - Pickup receiver
  - Stop ringing
Sequence diagram: with object creation, destruction, conditions

- obj1:C1
  - \([x > 0]\) create(x)
  - condition
  - object created by operation
  - object already existing

- obj2:C2
  - \([x \leq 0]\) m1(x)

- obj3:C3
  - object created by operation
  - object destroys itself and control returns to caller
Sequence diagram: with threads of control

- `obj1:C1` with `op1`
  - `[x>0] create(x)
  - `[x<=0] m1(x)

- `obj2:C2` with `obj3:C3`

- `obj4:C4`
  - Branch off control
  - Merge control
Sequence diagram: with self call

- op1
  - obj1:C1
    - [x>0] create(x)
  - obj2:C2
    - [x<=0] m1(x)
  - obj3:C3
  - obj4:C4
  - selfcall()
Interaction Diagram: Issue Item


- supply bid, uid
- create issue(bid, uid)
- bind using bid
- isIssuable?
- hasToken?
- issue (u)
- issue (b)
- destroy
- committed
Modeling Persistent Objects

The object has been created by Process1, but lives after Process1’s lifetime.
A State Machine for the Bounded Buffer Problem

- **Full**
  - Insert: $\text{size} = \text{MAX}-1$
  - Fetch: $\text{size} > 1$

- **Empty**
  - Insert: $\text{size} < \text{MAX}-1$
  - Fetch: $\text{size} = 1$

- **Partial**
  - Insert: $\text{size} = \text{MAX}-1$

Initial state: **Empty**
Handling Error Conditions

Initial state

- **Empty**
  - fetch / signal error
  - insert [SIZE = MAX-1]

- **Full**
  - fetch [SIZE = 1]
  - insert / signal error
  - insert [SIZE = MAX-1]

- **Partial**
  - fetch [SIZE > 1]
  - insert [SIZE < MAX-1]
  - fetch [SIZE = 1]
Actions in States

Busy

Entry / output
BusyTone ()

Exit / stop
BusyTone ()
Actions on Transitions

Empty

Partial

insert [SIZE < MAX] /
addElementToRear,
incrementSize,
updateFront

/ initialize
size to 0,
MAX to
capacity

Initial
An Activity Model for admission time formalities

- Pay Fees
  - Get photographed
    - collect I-Card
  - Collect Hostel Keys
    - Occupy hostel room
  - attend welcome function
  - open local bank account
References

• UML manuals & Specifications