Introduction to Object Oriented Modeling

R K Joshi

IIT Bombay
Modeling: Why?

R.K. Joshi, IIT Bombay
Modeling: Why?

Concepts to Realization
Modeling: Why?

Realization to Conceptual Understanding
Modeling: What?

Need to Model Different aspects of/for the software to be built
Modeling: What?

R.K. Joshi, IIT Bombay
Conceptual ideas

Modeling: What?

R.K. Joshi, IIT Bombay
Conceptual ideas

Modeling: What?

End goals

R.K. Joshi, IIT Bombay
Conceptual ideas

User interfaces

Modeling: What?

End goals

R.K. Joshi, IIT Bombay
Modeling: What?

Conceptual ideas

User interfaces

Product Implementation Models

End goals

R.K. Joshi, IIT Bombay
Modeling: What?

Conceptual ideas

User interfaces

Product Implementation Models

End goals

Deployment Models, N/W, Architecture

R.K. Joshi, IIT Bombay
Modeling: What?

Conceptual ideas

User interfaces

Product Implementation Models

Deployment Models, N/W, Architecture

End goals

Man month

R.K. Joshi, IIT Bombay
Modeling: What?

Conceptual ideas

End goals

User interfaces

Product Implementation Models

Deployment Models, N/W, Architecture

Man month

Financial Model

R.K. Joshi, IIT Bombay
Modeling: What?

Conceptual ideas

User interfaces

Product Implementation Models

Deployment Models, N/W, Architecture

The software development process

Financial Model

Man month

R.K. Joshi, IIT Bombay

End goals
Modeling

R.K. Joshi, IIT Bombay
Modeling

Modeling tools

R.K. Joshi, IIT Bombay
Modeling tools

Microscopic views & Iterative Refinement

HOW?

R.K. Joshi, IIT Bombay
Modeling

Modeling tools

Microscopic views & Iterative Refinement

Documentation

R.K. Joshi, IIT Bombay
Product Models

Structure

Dynamics

R.K. Joshi, IIT Bombay
Static Modeling

- Furniture?
- Air ducts?
- Rooms?
- Water pipelines
- Building?
- Electrical Wiring?
- Flats?

what aspects to model?

R.K. Joshi, IIT Bombay
Models throughout the Lifecycle

- Per class
- Per object
- Collaborating classes
- Collaborating objects
- Collection of classes (source code)
- Location of classes, modules ..
Structural (Static) Modeling

- Class diagrams
- Object Diagrams
- Deployment Diagram
- File directory hierarchy
- Component Package diagrams
- Entity Relationship
A Static Model

<table>
<thead>
<tr>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push()</td>
</tr>
<tr>
<td>Pop()</td>
</tr>
<tr>
<td>Size()</td>
</tr>
</tbody>
</table>
Dynamic Modeling

what aspects to model?

R.K. Joshi, IIT Bombay
Dynamic Modeling

How does the group collaborate?

what aspects to model?

R.K. Joshi, IIT Bombay
Dynamic Modeling

How does the group collaborate? How does the individual respond?

what aspects to model?

R.K. Joshi, IIT Bombay
What happens amongst the objects?
Interaction Diagram: *Item issue use case*

- **Operator**
- **iw:IssueWin**
- **tr1: Issue**
- **b:Book**
- **u:User**

**Actions:**
- `supply bid, uid`
- `create`
- `issue(bid, uid)`
- `bind using bid`
- `isIssuable?`
- `hasToken?`
- `issue (u)`
- `issue (b)`
- `destroy`
- `committed`
What happens inside an object?
A State Machine

Initial state

- **Full**
  - Insert: size = MAX-1
  - Fetch: size > 1

- **Empty**
  - Insert: size < MAX-1
  - Fetch: size = 1

- **Partial**
  - Insert: size = MAX-1
  - Insert: size < MAX-1

R.K. Joshi, IIT Bombay
Including Error Conditions

- **Initial state**: [SIZE = 1]
- From **Empty** to **Full**: [SIZE = MAX-1]
- From **Full** to **Empty**: [SIZE = 1] / signal error
- From **Full** to **Partial**: [SIZE = MAX-1]
- From **Partial** to **Empty**: [SIZE > 1]
- From **Empty** to **Partial**: [SIZE < MAX-1]
- From **Partial** to **Full**: [SIZE < MAX-1]

R.K. Joshi, IIT Bombay
How are various interactions/activities linked with each other?
Activity Diagram

Scan User’s barcode

[scan okay]  [scanner failed]

enter user id by hand

Scan Book barcode

[scan okay]  [scanner failed]

enter bookid by hand

Perform return transaction

R.K. Joshi, IIT Bombay
How much can you model?

- Can the full system be specified during modeling?
- What other modeling languages exist?
- Can the full system be generated from the model?
- Benefits of Modelchecking
- Modeling language vs. implementation language
Code Generation

• Class structures
• Basic Relations
• Basic constructions
• Helper Code
• Service Orientation
  – Intermediate formats and Platform dependent code generation techniques
Traceability

- Models
  - Seamlessness
- Moving from early conceptualization to construction and eventually into delivery and maintenance
- Changes reflected back
Summary

• Model ---> Implementation
• Implementation --> Model
• Forward engineering
  – Specify/model first and then go on to build
• Many facets of a system --> many models
• Consistency between models
• Traceability to Implementation
• Automatic Code Generation