CS 846: Model-Based Software Engineering
Winter 2012

Jo Atlee

David R. Cheriton School of Computer Science
University of Waterloo
January 16, 2012

Software Models

http://se.uwaterloo.ca/~jmatlee/teaching/846
What is Being Modelled?

 postal-address ::= <name-part> <street-address> <zip-part>

 <name-part> ::= <personal-part> <last-name> <opt-jr-part> <EOL>

 E = a KLOC
Models as Views

Every view

- obtained by a different projection, abstraction, translation
- may be expressed in a different notation (modelling language)
- reflects a different intent

[ Example from J. Bezivin, J. Dingel ]
Modelling Paradigms

Fundamental *modelling paradigms*, each emphasizing some basic view of the software to be developed.
Entity-Relationship Diagrams

An **ER diagram** is a *structural model* representing a software system's data elements and relationships among them.

- originally invented for model database design (Chen, 1976)
- emphasizes concepts/data
- relationships can represent associations, navigability, containment, dependencies, etc.
UML Class Diagrams are an elaborate form of ER diagram.

[ Figure from Pfleeger, Atlee, 2009]
A software architecture is a high-level model of code structure.

Typically modelled as a "box and arrow" diagram, with a key explaining the types of components (boxes) and connectors (arrows).

[ Figure from Pfleeger, Atlee, 2009]
UML "Software Architecture" Models

The closest that UML comes to a software architecture model are UML Package Diagrams and UML Deployment Diagrams.

UML Package Diagram

UML Deployment Diagram
Data Flow Diagrams (DFDs)

*Descriptive* model of *functional decomposition* of the system, and the data dependencies between functions.

DFDs model
- collection of functions
- sources and sinks of data
- data dependencies
Data Flow Diagrams (DFDs)

Although DFDs are good for communicating the big picture, they are inherently incomplete, undetailed, and ambiguous.

[ Figure from M. Jackson, 1995 ]
Use Case Diagrams are a very high-level data-flow diagram.

### Online Book Store

- **Customer**
  - Making Purchase
  - Managing Shopping Cart
  - Managing Customer Info
  - Log In
- **Administrator**
  - Managing Inventory
- **Bookseller**
  - Bookseller Inventory
- **Credit Authorization Service**

**Roles**
- **Actor**
- **Function**
Flowcharts are an ancient modelling notation, for representing *behaviour* in terms of steps of an algorithm.

Depict control flow rather than data flow
UML Activity Charts are a variation on flowcharts that support concurrent flows of control.
**Event Traces**

*Dynamic model of behaviour* showing communication among entities in one scenario (execution trace).

Shows a slice of behaviour, not complete behaviour.
**UML Sequence Diagrams**

**UML Sequence Diagrams** are elaborate event traces....

![Diagram showing a sequence of interactions between entities.](image)
... including branches, loops, concurrency, optional subsequences, references to other sequence diagrams.

Larman, *Applying UML and Patterns, 3ed*
State Machines

Compact representation of all event traces.
UML StateMachine Diagrams borrow heavily from David Harel's statecharts.

VARIABLES
req1 : boolean := false (* outstanding request for floor1 *)
req2 : boolean := false (* outstanding request for floor2 *)
floor : {1, 2} (* current location of elevator *)
Logic is the basis for a number of languages that express constraints on allowable interpretations of other models.

- allowable instantiations of data models
- invariants among attribute values in data models
- pre/post conditions of functions
- event conditions in event traces
- guard conditions in state machines
**Object Constraint Language**

**OCL** was designed for expressing constraints on UML diagrams.

**context** `CustomerCard` **inv**

self.printedName = (self.owner.title.concat(self.ownernname))

**context** `CustomerCard` **inv**

(self.valid and self.colour=gold) implied self.membership.serviceLevel = "gold"
UML: 13 Different Diagram Types

Structural
- Class Diagrams
- Object Diagrams
- Composite Structure Diagrams
- Component Diagrams
- Package Diagrams
- Deployment Diagrams

Behavioural
- Interaction Overview Diagrams
- Activity Diagrams
- Sequence Diagrams
- Communication Diagrams
- State Machine Diagrams
- Timing Diagrams

Functional
- Use Case Diagrams

OCL is a separate language that was invented for writing constraints on UML models
3 Prominent Modelling Languages

• Alloy
• Statecharts
• Scala
References


References


