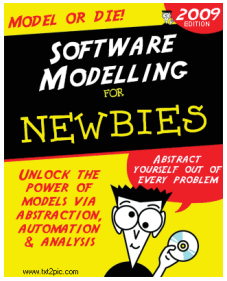
Beyond Code:
An Introduction to Model-Driven Software Development (CISC836)



Languages: UML

Juergen Dingel
January 2021

Expressing SW models: Overview

Examples of software modeling languages

1. **UML** (for modeling everything)
   - language: collection of 14 diagram types
   - analysis: e.g., well-formedness, approaches to consistency, reachability
2. **UML-RT (for soft real-time embedded)**
   - language: much smaller, domain-specific subset of UML
3. **Stateflow/Simulink (for control systems)**
   - language: domain-specific combination of statemachines and dataflow
4. **SMV, Promela** (for concurrent systems)
   - language: concurrent, imperative language with message passing
   - analysis: temporal logic model checking (i.e., exhaustive state space exploration) using NuSMV, Spin

Lots more:

Petri nets, queuing networks, synchronous languages (e.g., Lustre/SCADE), …

Modeling Languages

Modelica
- Physical systems
- Equation-based

Simulink
- Continuous control, DSP
- time-triggered dataflow

Stateflow
- Reactive systems
- Discrete control
- State-machine-based

AADL
- Embedded, real-time

UML

UML MARTE
- Embedded, real-time

Lustre/SCADE
- Embedded real-time
- Synchronous dataflow

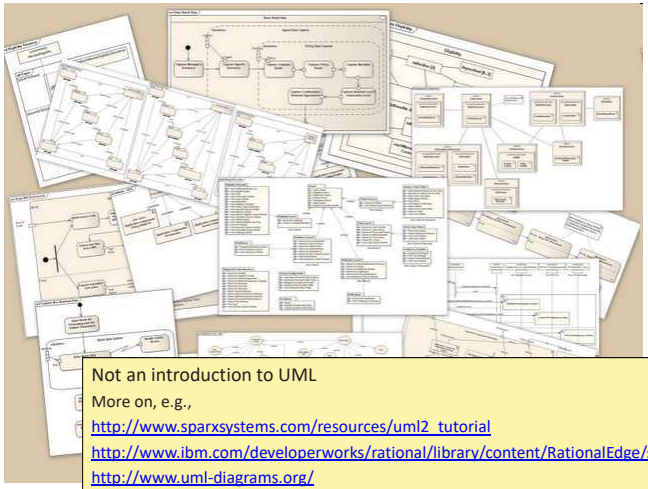
UML-RT
- Embedded, real-time
- State-machine-based

Examples in [Voe13, Kel08]

EGGG [Orw00]

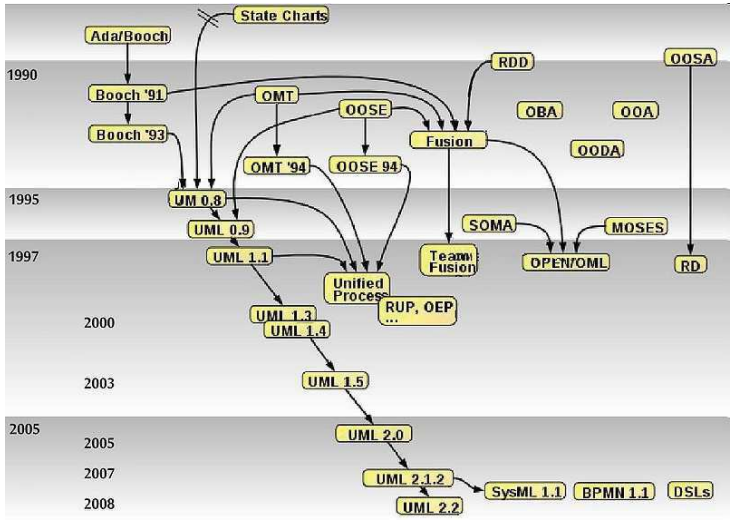
increasing generality ⟷ **increasing domain-specifity**

UML: A brief overview



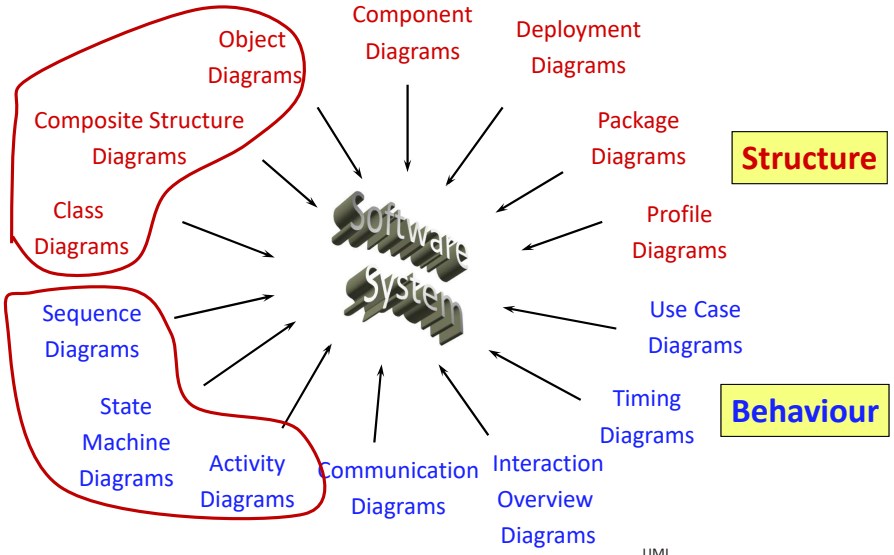
Not an introduction to UML
More on, e.g.,
http://www.sparxsystems.com/resources/uml2_tutorial
http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/
http://www.uml-diagrams.org/
http://www.omg.org/spec/UML/2.5/Beta2/PDF
http://link.springer.com/chapter/10.1007%2F978-3-642-30982-3_1

UML: History

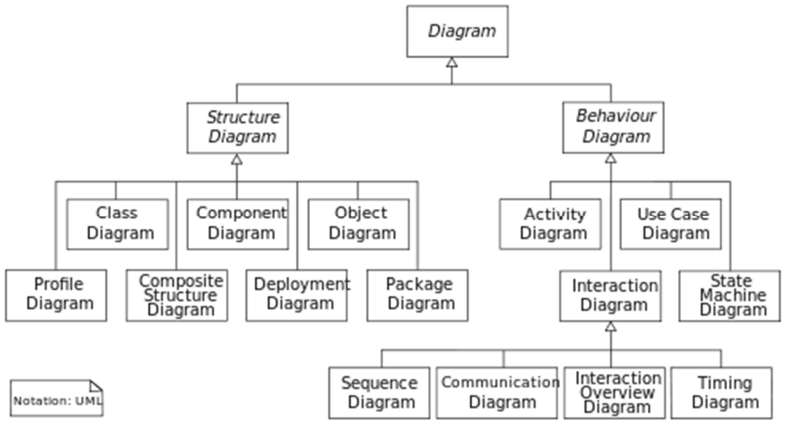


http://en.wikipedia.org/wiki/Unified_Modeling_Language

UML: 14 Different Diagram Types

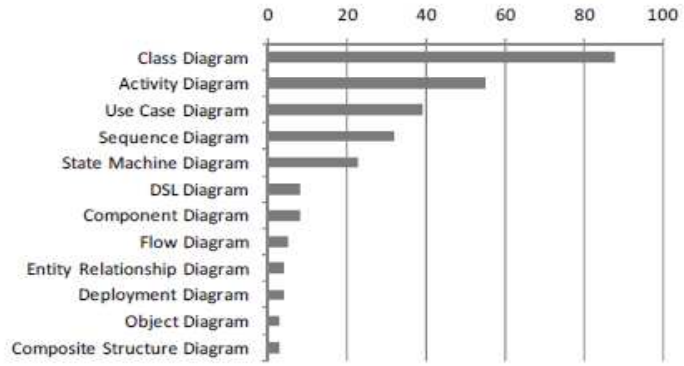


UML: 14 Different Diagram Types (Cont'd)



UML: Class Diagrams

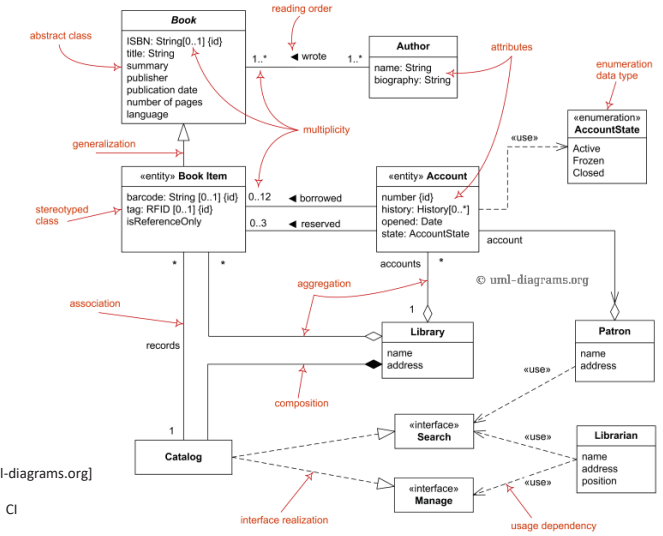
- Widely considered most important and most useful [ES07, Whi11a]
- Forms basis of many language specification techniques (MOF, Ecore)



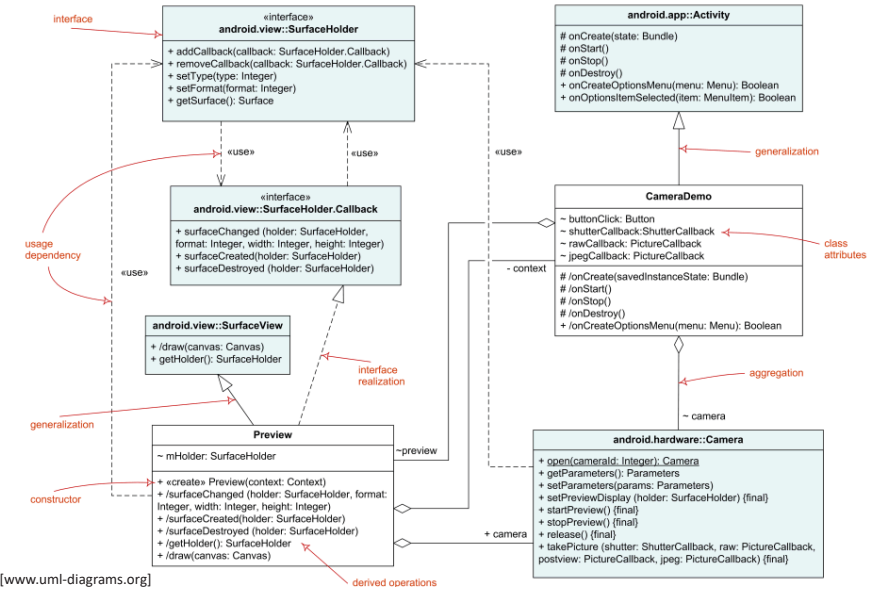
[Whi11a] Hutchinson, Whittle, Rouncefield, Kristoffersen. Empirical assessment of MDE in industry. ICSE'11. 2011

UML: Class Diagrams (Cont'd)

Shows classes/concepts, their attributes, operations & relationships



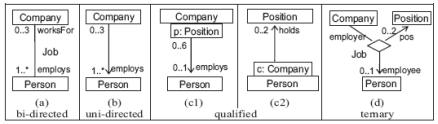
[www.uml-diagrams.org]



[www.uml-diagrams.org]

UML: Class Diagrams (Cont'd)

- Associations are a rich concept
  - Multi-arity, multiplicity, navigability, visibility, ownership of ends (by classifier or by association), qualification, association classes

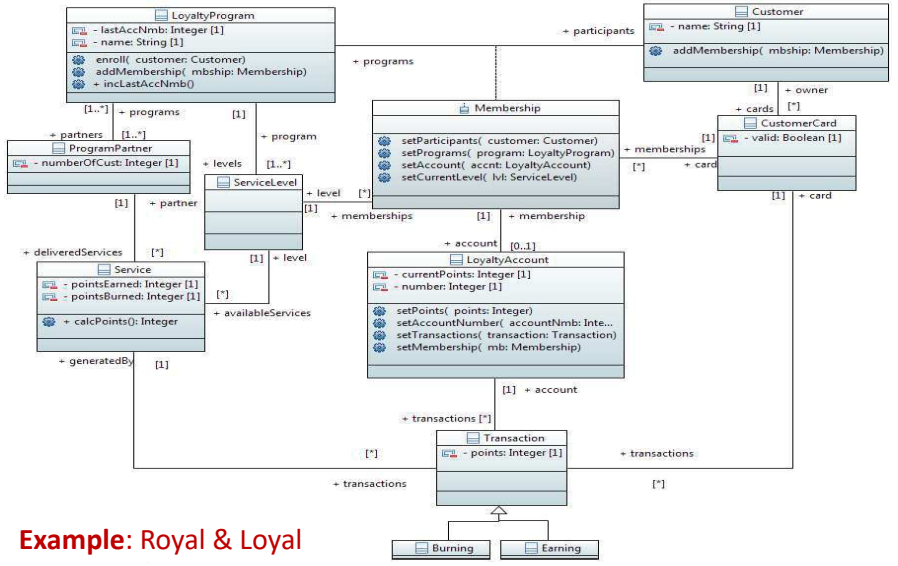


- Code generation not straight-forward
  - E.g.,
    - checking of multiplicity constraints
    - if both ends of a binary association are navigable and owned by the end classes, then update of one association end may require update of other as well [Ges08]

```
1  public class A{
2     private B b;
3     public boolean setB (B value){
4        if (this.b == value) return false;
5        B oldValue = this.b;
6        this.b = value;
7        if (oldValue != null)
8           oldValue.setA (null);
9        if (value != null)
10          value.setA (this);
11       return true;
12    }
13    public B getB () { return this.b; }
14 }
```

Listing 1. Implementation of mutual updates for links of an association

UML: Class Diagrams (Cont'd)

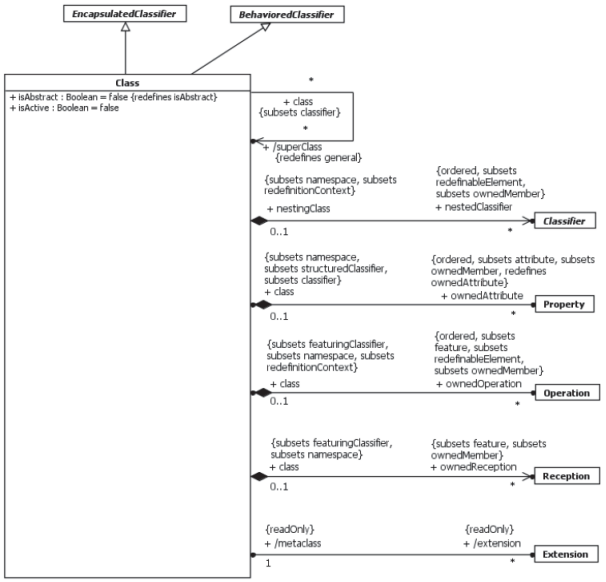


Example: Royal & Loyal

UML: Class Diagrams (Cont'd)

- **Examples**:
  - Software design patterns
  - http://en.wikipedia.org/wiki/Software_design_pattern
    (e.g., Factory, Composite, Proxy, Observer, Visitor)

- But what really is a class diagram? Can we use a class diagram to describe the concepts that make up a class diagram?

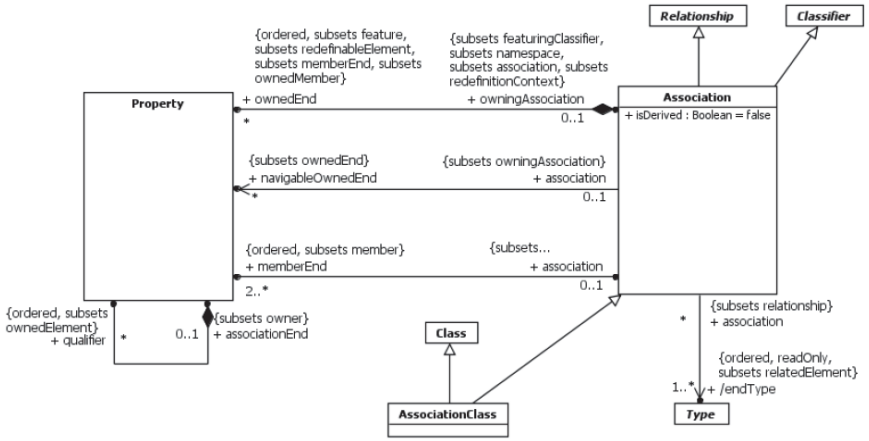


UML: Class Diagrams (Cont'd)

Example:
Classes in UML 2.5 Specification [Section 11.4, page 202]

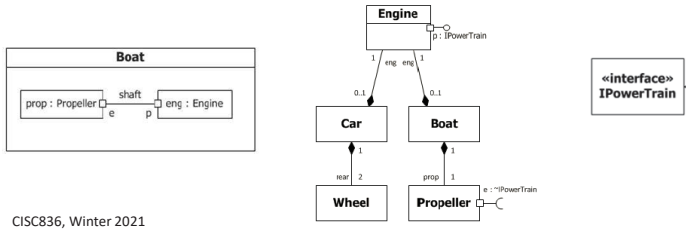
UML: Class Diagrams (Cont'd)

Example: Associations in UML 2.5 Specification [Section 11.5, page 208]

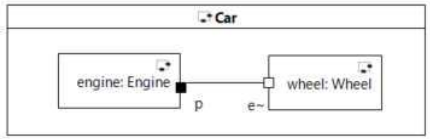


UML: Composite Structure Diagrams

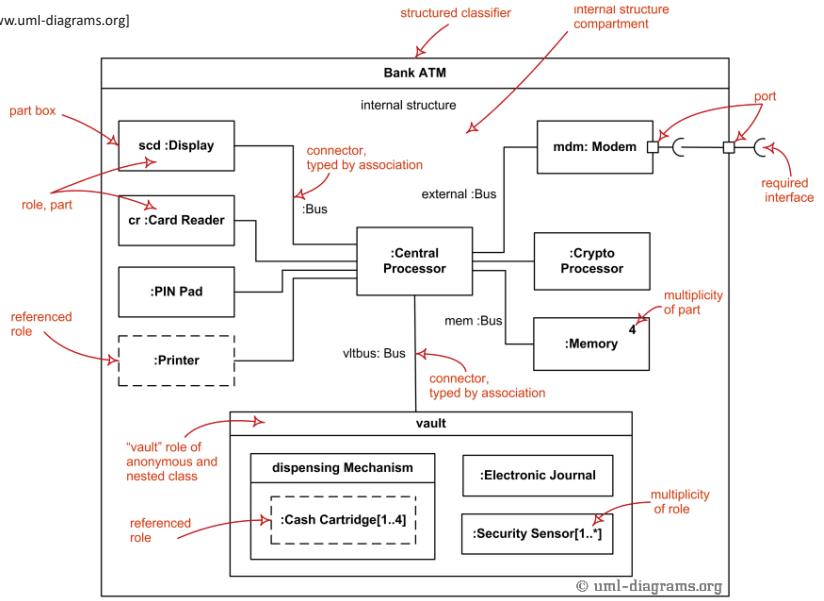
- Shows internal structure of StructuredClassifier, including interaction points to other parts of system
- **Key concepts**
  - **Part:** Properties specifying instances that StructuredClassifer owns (i.e., properties w/ aggregrationKind=composite)
  - **Port:** typed element defining interaction between classifier and environment; may specify provided and required services (via **interfaces**)
  - **Connector:** specifies links between parts; typically represents possibility to communicate; typed by Association



UML: Composite Structure Diagrams (Cont'd)

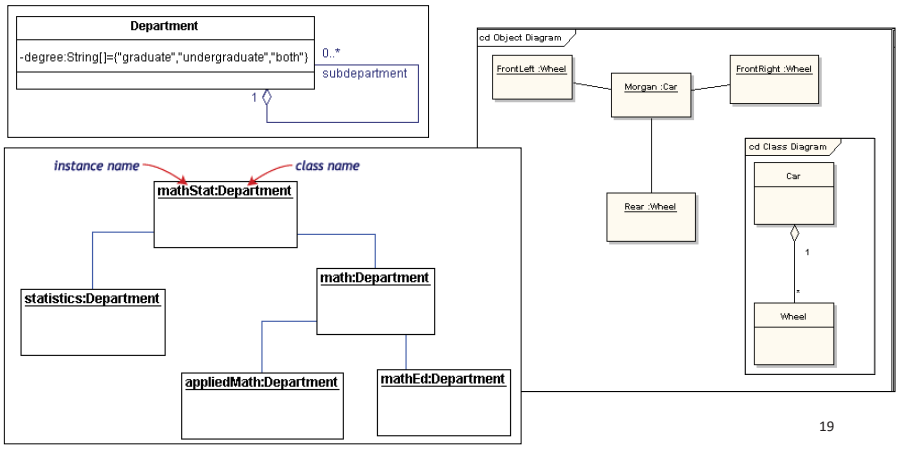


[www.uml-diagrams.org]

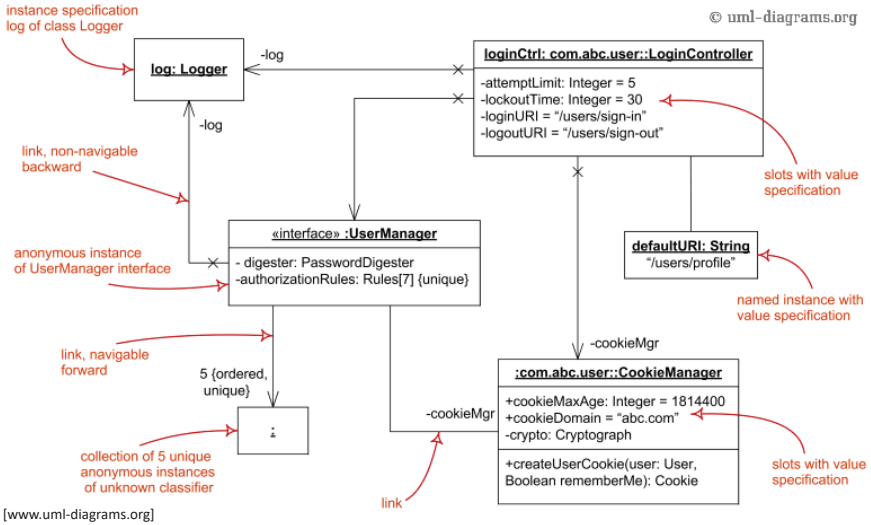


UML: Object Diagrams

- Shows objects/instances and their relationships at particular point in time (a.k.a., "snapshot" or "state")
- Must be conforming to class diagram



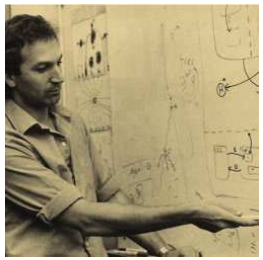
UML: Object Diagrams (Cont'd)

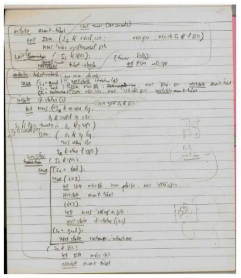


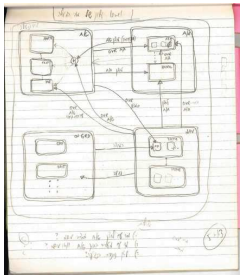
[www.uml-diagrams.org]

UML: State Machines

David Harel





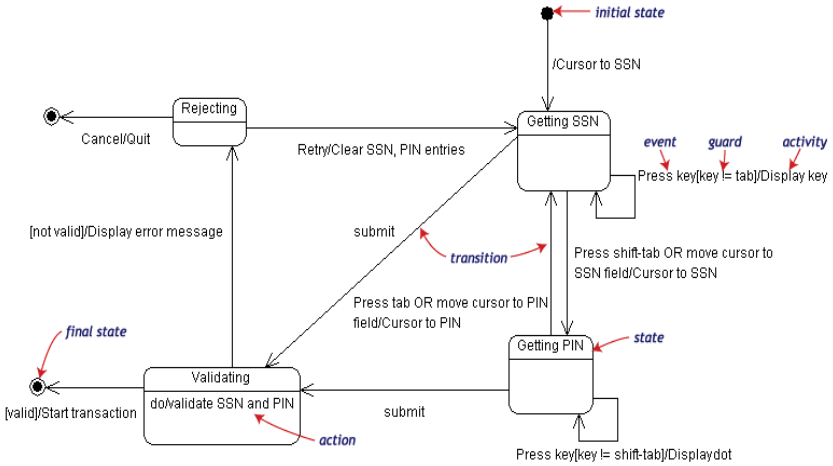


> *"The pictures were simply doing a much better job of setting down on paper the system's behavior, as understood by the engineers, and we found ourselves discussing the avionics and arguing about them over the diagrams, not the statocols."* [Har07]

[Har07] D. Harel. Statecharts in the Making: A Personal Account. 3rd ACM SIGPLAN Conference on History of Programming Languages. 2007.

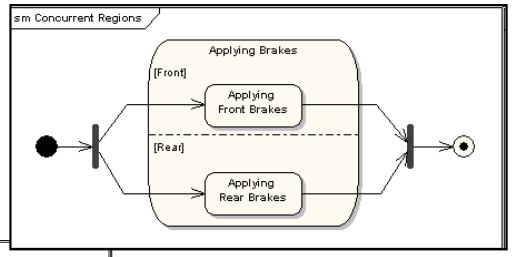
UML: State Machine Diagrams

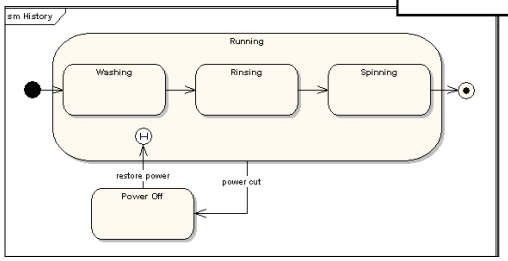
Show behaviour as sequences of state changes caused by transitions triggered by events

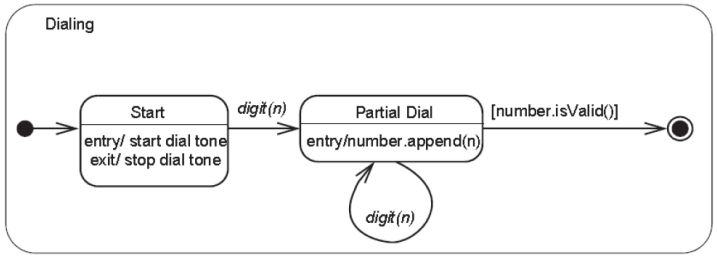


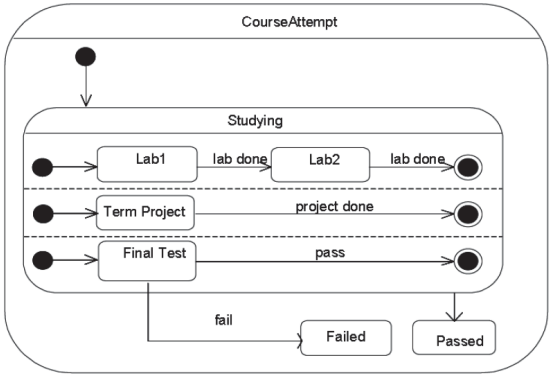
UML: State Machine Diagrams (Cont'd)

- **Features**
  - Composite states (hierarchical, or-states)
    - Group transitions
  - History states
  - (Orthogonal, concurrent) regions (and-states)
  - Entry, exit, do-actions



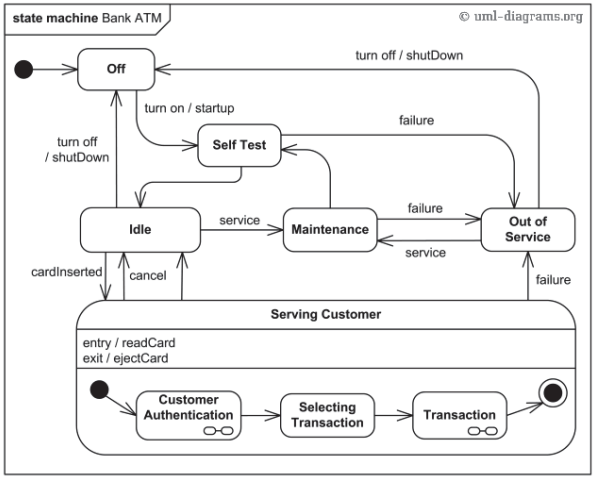






UML: State Machine Diagrams (Cont'd)

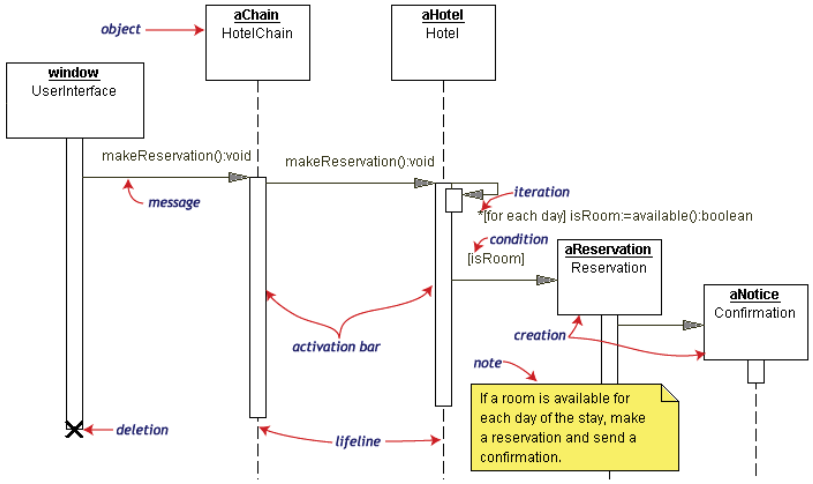
UML: State Machines (Cont'd)



[www.uml-diagrams.org]

UML: Sequence Diagrams

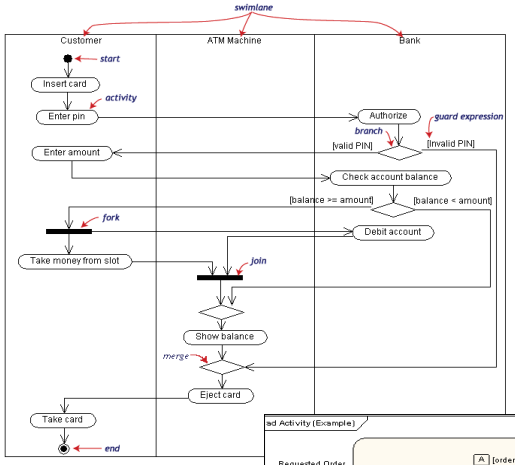
- Show behaviours as sequences of messages b/w objects

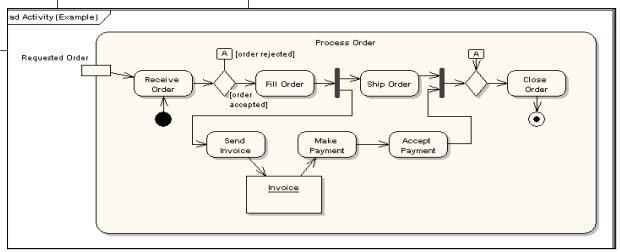


UML: Activity Diagrams

- Show behaviours as sequences of activities
- **Features**
  - **Two kinds of flow**: control and data
  - **Different kinds of behaviour invocation**: synchronous, asynchronous
  - **Different kinds of control nodes**: initial, final, fork, join, decision, merge
  - **Different composition mechanisms**: loops, conditionals, interruptible regions, exceptions
  - **Structuring mechanisms**: partitions, swimlanes
  - **Support for data flow**: edge weights, multiplicities on pins
- **Semantics:** Petri net-based "token/offer" semantics with deadlock avoidance rules

UML: Activity Diagrams (Cont'd)





Object Constraint Language (OCL)

- A declarative language for describing well-formedness rules of models
- May be used with any class diagram
- **Examples:**
  - *"The source & target states of transition belong to same machine"*

    Transition

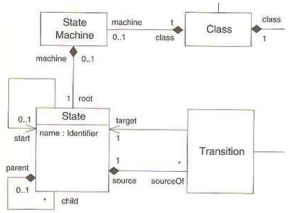
    target.root().machine = source.root().machine

    where **root()** is

    State::root() : State {

    if parent = null then self else parent.root()

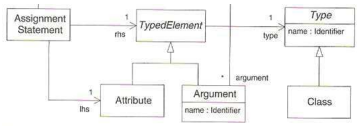
    }



  - *"The left-hand side and the right-hand side of an assignment have the same type"*

    AssignmentStatement

    lhs.type = rhs.type



UML: Tools

- Commercial
  - RSA, RSARTE, Rhapsody (IBM)
  - MapleMBSE (Maplesoft)
  - MagicDraw, Cameo (No Magic)
- Open source
  - Papyrus
    - eclipse.org/papyrus
  - Papyrus for Information Modeling (for class diagrams)
    - https://wiki.eclipse.org/Papyrus_for_Information_Modeling
  - Mentor Graphics xtUML
    - http://www.xtuml.org/
  - USE (for OCL)
    - sourceforge.net/apps/mediawiki/useocl
- Web-based
  - Draw.io

UML: Summary

- De facto standard in software modeling
- Rich "dictionary" of model concepts
  - UML 2.5 Spec has 809 pages
  - *"UML was designed to be used selectively"* Bran Selic in [Pet14]

  ⇒ best to approach study of UML with particular purpose, need
- Tool support
  - Still a problem, but getting better
  - Increasingly open source