System Models - Behavioral

- Requirements and Behavioral Models
- the Universal Modeling Language
- UML behavioral models
  - interaction diagrams
  - activity diagrams
  - swim-lane diagrams
  - state diagrams
- non-UML behavioral models
  - data flow diagrams

Behavioral Requirements

- Use Cases describe system capabilities
  - in an abstract, problem-centric way
  - use cases describe what, not how
  - they are usually incomplete as requirements
- We need a behavior description language
  - to describe the individual steps of a task
  - to describe the interactions in each step
  - to better capture the user's needs
  - to provide clearer guidance to developers

Requirements vs. Specifications

- Start w/user-level functional requirements
  - expressed in terms of user experience
  - what the system can do, how it will behave
- Add non-functional requirements
  - structural, stylistic, technological, components
- Project them onto an architecture
  - associate requirements w/each component
  - what this component is capable of doing
  - how it will behave in specified situations

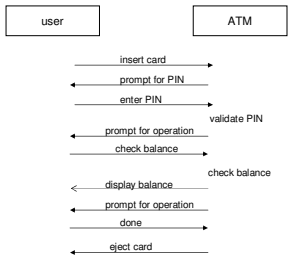
Universal Modeling Language

- a family of related graphical notations
  - for representing software system designs
  - particularly those built in object oriented style
- supports a wide range of uses
  - a design sketching language
  - a system specification language
  - a programming language
- also has a textual (XML) representation
  - enabling development of CAD tools

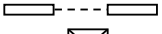
UML Interaction Diagrams

- Describe interactions
  - between multiple actors
  - between actors and the system
  - typically one diagram per task or scenario
- Simple and intuitive representation
  - easy to draw, easy to understand
- Excellent for behavioral requirements
  - illustrative sample usage scenarios
  - additional detail for a use-case or story card

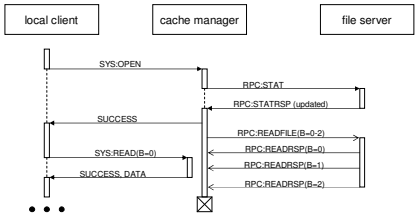
UML User Interaction Diagram



Advanced Interaction Diagrams

- Describe system component interactions
  - collaborations between objects
  - remote procedure calls and messages
- Rich vocabulary for describing interactions
  - descriptions of messages and requests
  - synchronous (procedure call)
  - asynchronous (message)
  - active/blocked threads 
  - thread creation and destruction 

UML Object Interaction Diagram



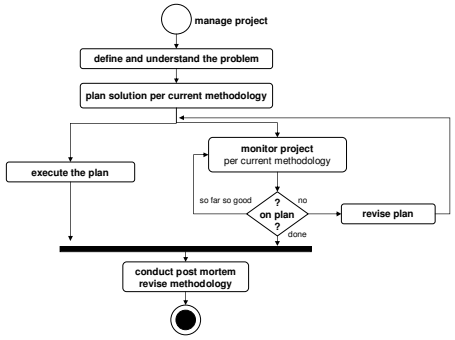
Using UML Interaction Diagrams

- Interaction Diagrams show interactions
  - users interacting with system components
  - interactions between system components
  - collaborations between object instances
- They can be used descriptively
  - to illustrate how a system will work
- They can be used prescriptively
  - to define expected behavior
- They are not for expressing algorithms

UML Activity Diagrams

- describe the steps of a computation
- circles represent starts and ends
- rounded boxes represent actions
  - at any desired level of abstraction
- diamond boxes represent decisions (ifs)
  - labeled arrows represent the decision cases
- arrows represent flow of control
  - diverging arrows for parallel activities
  - horizontal bars for join-points

UML Activity Diagrams



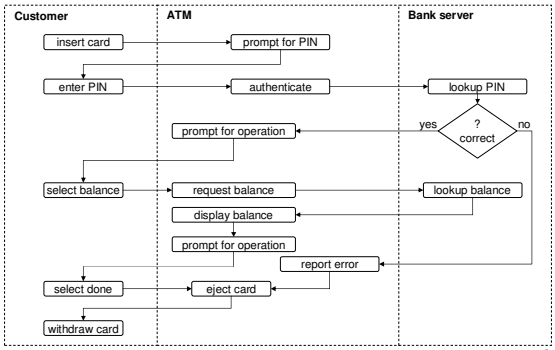
Using UML Activity Diagrams

- Activity Diagrams show procedural logic
  - like a flow chart, with addition of parallelism
- they can be used free-standing
  - as a process or algorithm description
- they can be placed inside of an object
  - attached to the associated interfaces
- there is a notation for procedure calls
  - method to call can be show, parenthetically, underneath the general action description

UML Swim-Lane Diagrams

- combine interaction and activity diagrams
- describe multi-threaded flow of control
  - remote procedure call and return
  - asynchronous message exchanges
- parallel threads in parallel columns
  - each with its own activity diagram
- horizontal lines represent messages
  - from sender to receiver
- horizontal bars represent joins
  - awaiting reception of a message

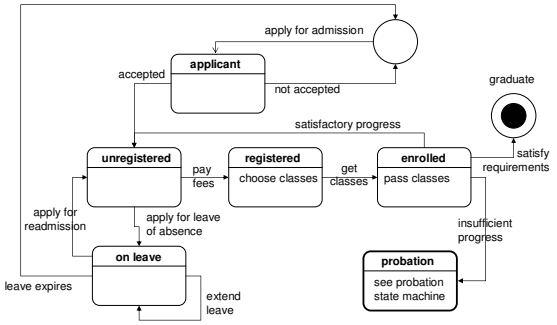
UML Swim-lane Diagrams



UML State Models

- describe state/transition models
  - where events drive state changes
- similar to activity diagrams
  - activity boxes have two compartments
    - state name in the top portion
    - processing steps in the bottom portion
  - arrows represent state transitions
    - from previous state, to next state
    - labels describe conditions triggering the transition
    - processing steps can also be placed on lines

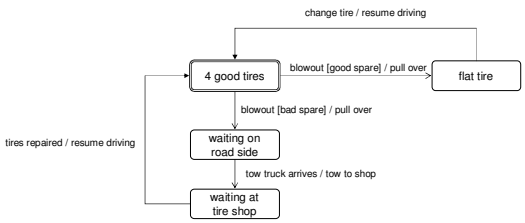
UML State Diagrams



Finite State Machine Transitions

- UML defines three parts to an arc label
  triggering event **[** *guard condition* **]** / *action*
- Where
  - *triggering event* is the event that will cause this transition
  - *guard condition* is a boolean test that determines whether or not this arc will be followed
  - *action* is an action that the system will take before entering the next state.
- These make it possible to directly translate traditional finite state machines in UML

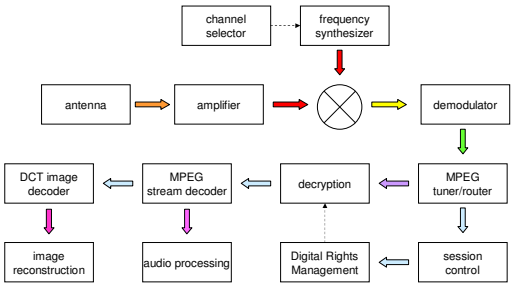
Finite State Machine



Data Flow Models

- UML is designed for OO-software
  - components are comprised of related objects
  - objects have properties and methods
  - methods have associated algorithms
  - interactions are via discrete messages
- Data Flow Models take different view
  - follow input, through processing, to output
  - all components are processing in parallel
  - processing is continuous rather than discrete
  - view system in terms of data transformations

Digital TV Data Flow Model



So many models …

- There are many models to choose from:
  - use case diagrams
  - interaction diagrams
  - activity and swim-lane diagrams
  - state diagrams
  - data flow models
- Each shows very different things
  - which one should we choose?

For Next Lecture

- Rational: UML Class diagrams
  - good overview of the richest part of UML
- Agile Modeling: UML Object diagrams
  - quick intro to diagramming object instances
- Agile Modeling: UML Package diagrams
  - quick intro to a very simple diagram
- Braun: Component/Deployment diagrams
  - quick intro to a very simple diagram