System Modeling

- Classes and levels of system models
- General principles of modeling
- Analytical models
  - algebraic, probabilistic, combinatoric
  - queuing models
  - Markov models
  - discrete event simulations
- Prototyping projects
- Executable Designs

Models

- are smaller and simpler than the real thing
  - making them less expensive to build
  - making them more portable
  - making them easier to understand
- often model only subsets of whole system
  - stripping away layers complicating details
  - permitting system to be understood in parts
- may expose otherwise invisible processes
  - making those processes easier to understand
- are only approximations of reality
  - they may lead to inaccurate predictions

General Types of Models

- Descriptive models
  - built to facilitate communication
    - help users understand what will be built
    - help developers understand what to build
- Analytical Models
  - built to answer questions or reduce doubt
    - clarification and validation of requirements
    - questions about a proposed implementation
- Mock-ups and "Proof-of-Concept"s
  - built to sell an idea
    - convince someone that the idea will work

General Modeling Principles

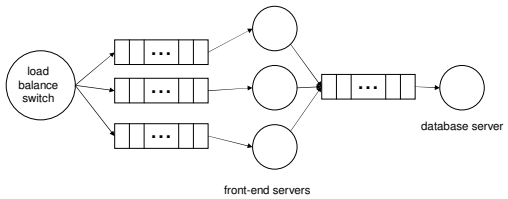
- Model with a purpose
  - be clear why you are building each model
- Travel light
  - maintain as few models as possible
  - know which (few) models are keepers
- Use multiple models
  - don't try to make one model serve all needs
- Content is more important than format
  - a good form is one that achieves your goals

Simple Mathematical Models (just do it)

- Algebraic Models
  - make up equations to describe situations
  - don't be afraid to estimate parameter values
    - use ranges and independent estimates
- Probabilistic Models
  - estimate event likelihood, frequencies
  - outcome expectancies (*cost x probability*)
- Combinatoric Models
  - how many possible combinations are there

Queuing Models

- Mathematical models of traffic and servers

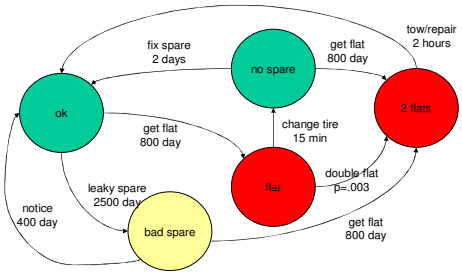


Queuing Models (don't try these at home)

- Given:
  - a system of input queues and servers
  - request arrival and processing rates
- Assuming:
  - arrivals have a *standard* distribution
  - processing time is same for all events in queue
- Model will yield closed-form solutions for:
  - queue length (distribution function)
  - waiting time (distribution function)

Markov Availability Models

- Mathematical models of state transitions



Markov Models (there are nice tools for these)

- Given:
  - a state machine representation of a system
  - w/mean transition rates (and/or probabilities)
- Assuming:
  - all events follow *standard* distribution
  - transitions have no memory of past events
- Model will yield numeric solutions for:
  - % of time system will spend in each state
  - mean visit duration for each state

Discrete Event Simulations

- simulate dynamic system behavior
  - for systems other techniques can't model
    - e.g. future events depend on past details
  - for questions other techniques can't answer
    - e.g. "Why are there so many cache misses?"
  - to exercise scheduling/routing algorithms
    - run simulated traffic through a real algorithm
- there are very abstract models
  - may be written in special simulation language
  - code may be very different from real system

For Next Lecture

- Usability.net: User Centered Design Principles
  - good set of basic principles
- Talin: Principles of U/I design
  - good discussion of philosophy of U/I design
- Nielsen/Butler: Web vs GUI design issues
  - how Web apps are similar and different from GUIs
- Withrow: diagramming web sites
  - introduction to a simple concept
- Bieber: Relationship Navigation Analysis
  - a methodical approach to content architecture
- Wikipedia: Usability Testing
  - good overview, ignore the stuff on statistics of sample sizes
- Kampe: Command Line Interfaces
  - there are actually principles to guide these interfaces as well

Supplementary Slides

Prototype to reduce Risk

- User Interface Prototypes
  - Do we have the U/I requirements right?
- Mechanism Prototypes
  - Do we know how to build this?
  - How well will it work?
- Process Validation
  - Do we really know how to process this?
- Tool and Platform Evaluation
  - How much trouble will this new stuff cause?

Proof of Concept

- You need help to succeed
  - managers, investors, customers
- They may be afraid to help you
  - if you fail, they will suffer losses too
  - they may doubt your ability to succeed
    - the proposed product might not be compelling
    - you might not be able to deliver it on schedule
    - there may be unsolvable development problems
  - these doubts must be assuaged
- well designed proof-of-concept can do it (D1)

Design Model Based Tools

- Design Visualization Tools
  - browse through hierarchies of models
  - selecting views
  - filtering displayed contents
- Design Validation Tools
  - style & standards conformance checkers
  - consistency checkers
  - interface based test case generators

Automatic Code Generation

- class model compilers
  - generate module skeletons
  - declarations for classes, methods, properties
  - stubbed routines to permit basic build/test
- rough code generators
  - simple code from activity diagrams
  - calls from interaction diagrams
- state language compilers (E)
  - generate final code from state machines