Lecture Overview

- What is a Model?
- Uses of Modeling
- The Modeling Process
    - Pose the Question
    - Define the Abstractions
    - Create the Model
    - Analyze the Data
- Model Representations
    - Queuing Models
    - Petri Nets
    - State Machines

What Is a Model?

- A representation of a system that is
  - *simpler than the actual system*
  - *captures the "essential characteristics"*
  - *evaluaable for a property of interest*
- *Examples*
  - A hardware system as a block diagram.
  - An operating system as a network of queues and servers.
  - A software system as an object model.
  - A parallel computer as a Petri net.
  - A communication protocol as a set of interacting state machines.

“Simpler”

- **Easier to construct than the actual system**
  - *server utilization = .005 * #users*
  - *Performance in 1 year can be estimated by running a workload generator on the current system*
- **“Simpler” often means “has fewer parameters”**
  - **Example: Computational Complexity (O(•) notation):**

    captures almost no details of any real hardware or implementation, yet provides useful information.

“Simpler” almost always means a more abstract representation than the representation of the actual system.

- **Example: A resource allocator as a queue/server pair.**

“Captures the essential Characteristics”

- **Includes only those characteristics that affect the property to the level of accuracy desired for the question that has been posed**
  - ***My web server needs more disk capacity. Can I just add disks or must I add servers?***
  - ***Do you model file buffer space effects or not?***

- **Deciding whether or not a model captures the essential characteristics is a multi-dimensional issue.**
  - **Formal Specifications and Proofs**
  - **Informal arguments, experience**

"Can be evaluated"

Values of metrics for the property of interest can be evaluated from the model.

- Example: Performance of a web server
  - What is metric for performance of a web server?
  - What kind of model will yield the desired metrics?
  - What are the available evaluation methods for the model representation?
  - How will I know the accuracy of the values of the metrics?
    - Validation?
    - Verification?

Example : Two processes are not writing a file at the same time.

- How specified?
- How verified?

What Is a Model?

- **A representation of a system that is**
  - * ***simpler*** **than the actual system**
  - * ***captures the essential*** **characteristics**
  - * ***can be evaluated for a specific property***

- **There is always an interaction among these three properties of a model.**
  - * **Faithfulness versus effort.**
  - * **Accuracy of evaluation versus effort.**

Uses of Modeling

- System Planning
  - Requirements
  - Constraints
- System Design
  - Choice of Components
  - Relationships among components
  - Performance
  - Availability
  - Correctness
- System Evaluation
  - Performance
    - Queuing Networks
    - Simulation Models
  - Correctness
    - Model Checking
  - Reliability
    - Stochastic Modeling

The Modeling Process for an existing system.

Pose the Questions
Define the Abstractions, Metrics and Parameters.
Create the Model
Gather and Analyze the Data
Measure the system and parameterize the Model
Verification/Validation
Explore the Parameter/Metric Space.

Modeling Process Example:

- **Imagine we have a parallel application running on a dual processor and want to project performance on larger machines**
- **Pose the Question/Define the Metric and the Parameters**

  What will the speedup be on 8 processors?

  Metric - Speedup

  Parameters - Number of Processors

  Are there hidden parameters?
- **Create the Model - Amdahl's Law**

  $S(f,N) = 1 / (f + (1-f)/N)$

  f is the serial fraction of the computation.

  N is the number of processors

Modeling Process Example Continued:

- **Measure the system and parameterize the Model**

  Suppose measured $S(2) = 1.79$. Then we deduce that $f = 0.12$.

- **Verification/Validation**

  Evaluation is not a problem.

  How good is the model?

    What factors are overlooked in Amdahl's Law?

- **Evaluate the Model for cases of interest**

  $N = 8$

  $S(8) = 1 / (0.12 + .88/8) = 4.35$

- **Sensitivity Analysis**

  What is sensitivity of model to f?

Pose the Question

The question posed strongly affects the kind of accuracy required for a performance study.

**** What will the response time be if the server is upgraded from a 500 MHz PII to a 933MHz PII?***

· The model must give a single number to within X% of true value.

**** How much faster will the 933 MHz system be than the current system?***

· If model underestimates response time by 20% on both the original and the projected systems, answer will still be exact

Pose the Question

∗ Is the 933 MHz system faster than the original?

• Suppose the model says "yes" and real system in fact has response times that are 80% of the original system.

Let α be the modeling error in the modified system and β be the modeling error in the existing system.

Modeled performance system projected
-- < 1
Modeled performance system existing

$(\alpha * 0.8) / \beta < 1$

$\alpha / \beta < 1.25$

Pose the Question

- For performance studies, comparing several alternatives to each other is much easier than other sorts of questions.

- On the other hand, you don’t know what the true answers are, and your confidence that the model is behaving accurately in a comparative sense typically decreases as your suspicion about the absolute error grows.

Create the Model

- ***"How is the model represented?"*** **versus** ***"How is the model evaluated?"***
- **These two are often intimately intertwined (and/or confused)**

Representation	***Property***	***Evaluation***
System similar to the projected system	**Performance**	***Measurement*** **or** ***Benchmarking***
Arbitrary set of equations.	**Performance**	***Analytic model***
Arbitrary software description of system behavior (state changes).	**Performance or Correctness**	***Simulation model***
Abstract state machine model	**Correctness**	***Model checking***

Create the Model

- **Higher-level description languages can be useful**
  - ∗ More convenient expression
  - ∗ Provide some structure for thinking about how to model the system
  - ∗ Determine what needs to be measured
  - ∗ "Canned evaluations"

- **As always, there is a trade-off between convenience and generality: the more tailored a model description language is to the environment, the easier it is to apply there, but the more difficult it is to use for anything else**

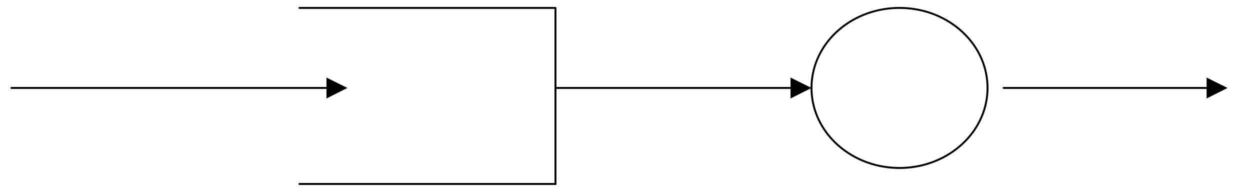
Model Description Representations

- We're going to look at three abstract representations of systems - queuing models, Petri nets and abstract state models.

- Only the briefest introductions here - just enough to have some common vocabulary until we look at these topics in more detail

Queuing Model Overview

- The basic component is the *queue/server pair.*



Customer Arrivals	Queue Server	Service Center	Customer Departures

Parameters

* Customer arrival rate
* Average service time
* Scheduling discipline
* ...

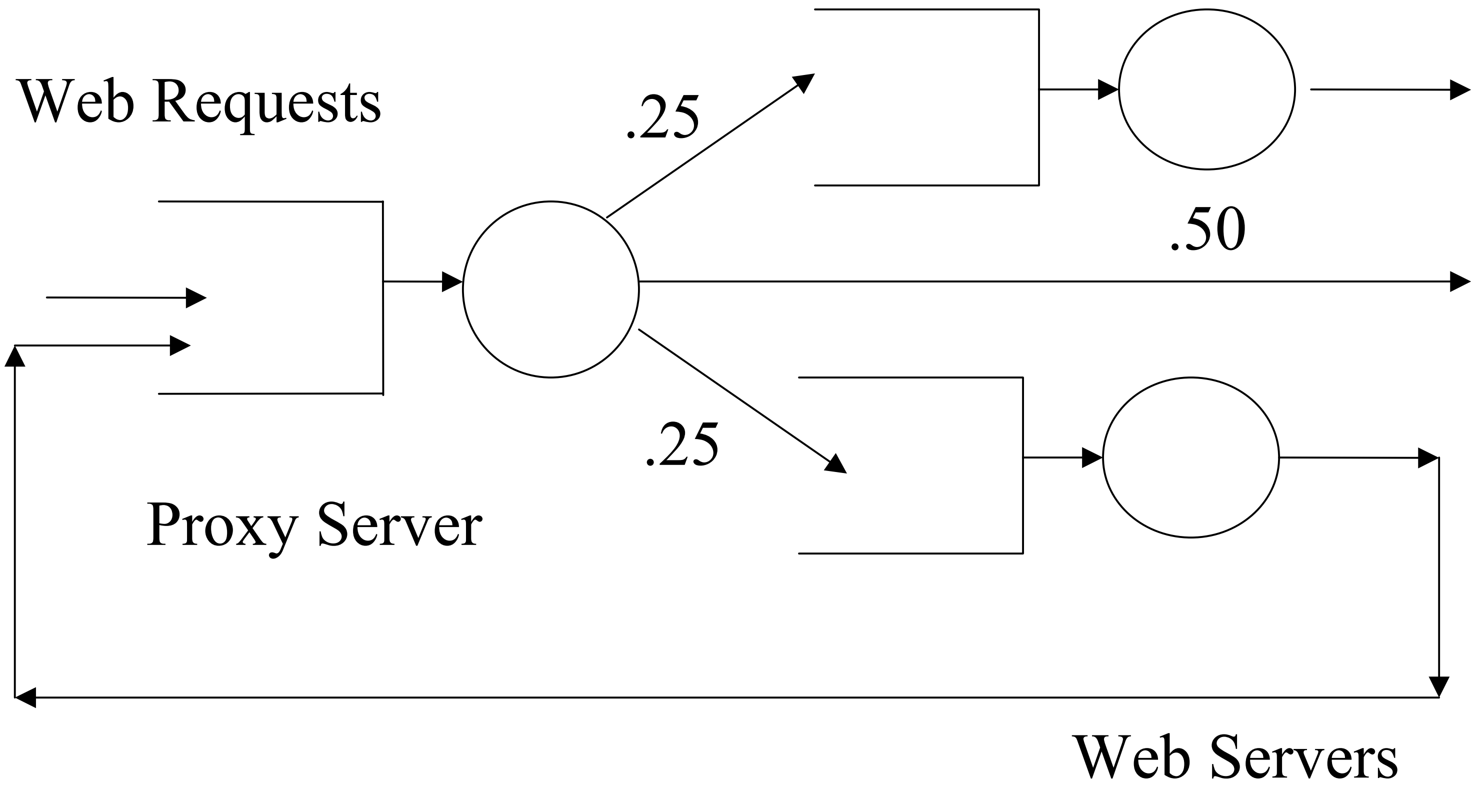
Intrinsic representation of time.

System State

* Number of jobs in queue

Queuing Model Overview

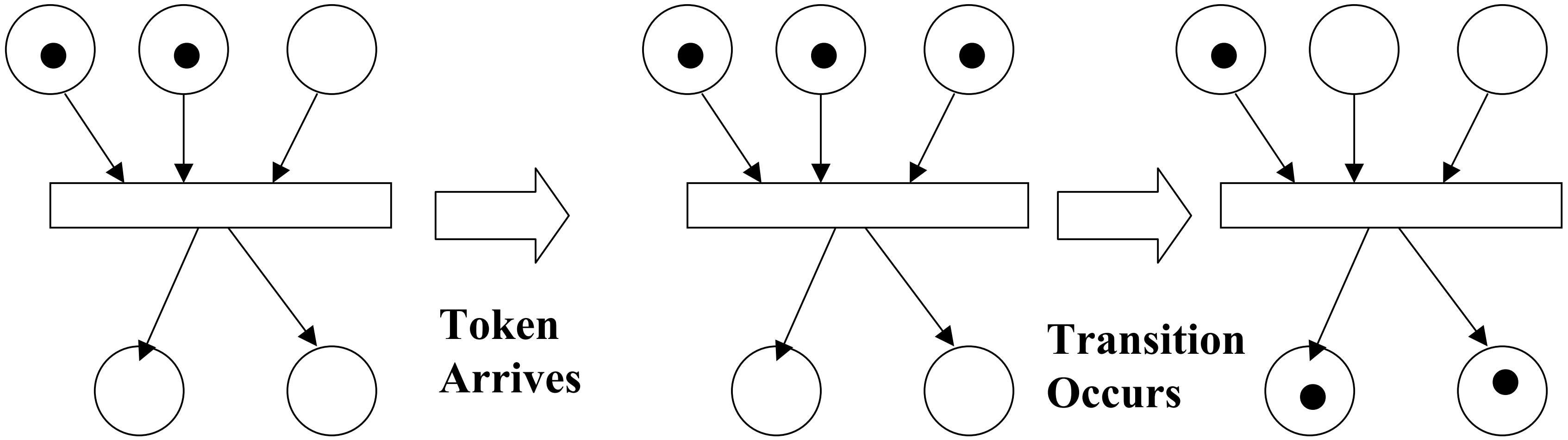
- We can construct networks of service centers



Open versus Closed, etc.

Petri Net Overview

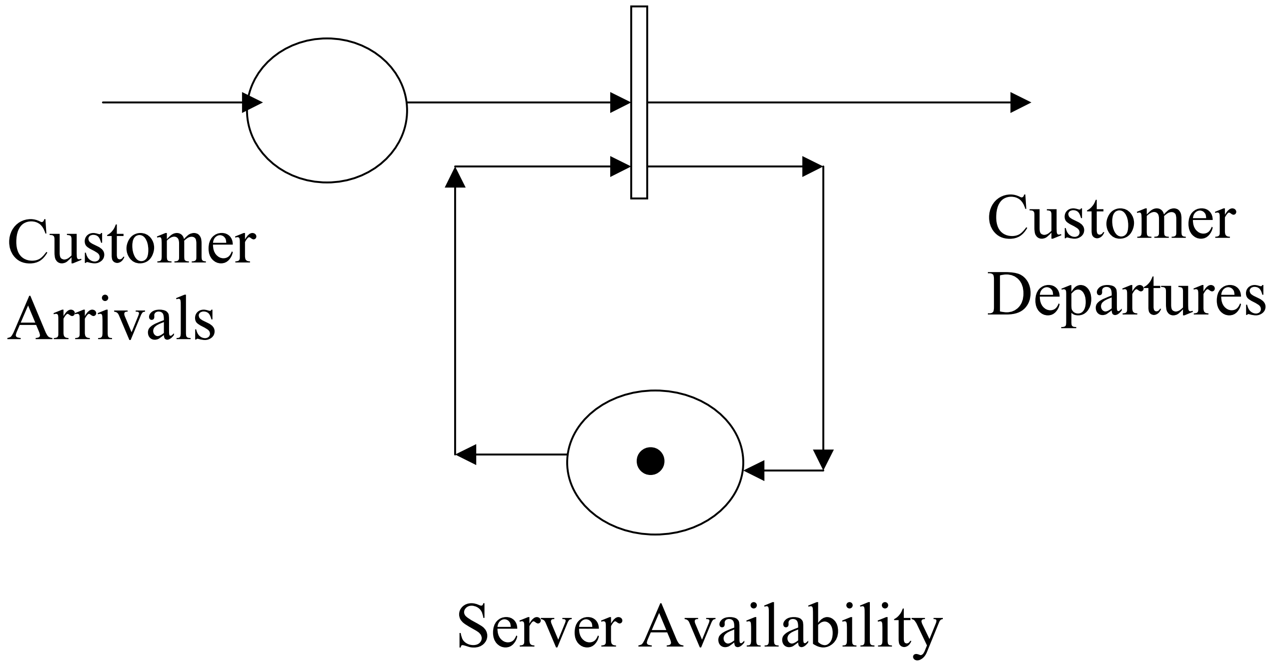
- Basic components are *places*, *transitions*, and *tokens*



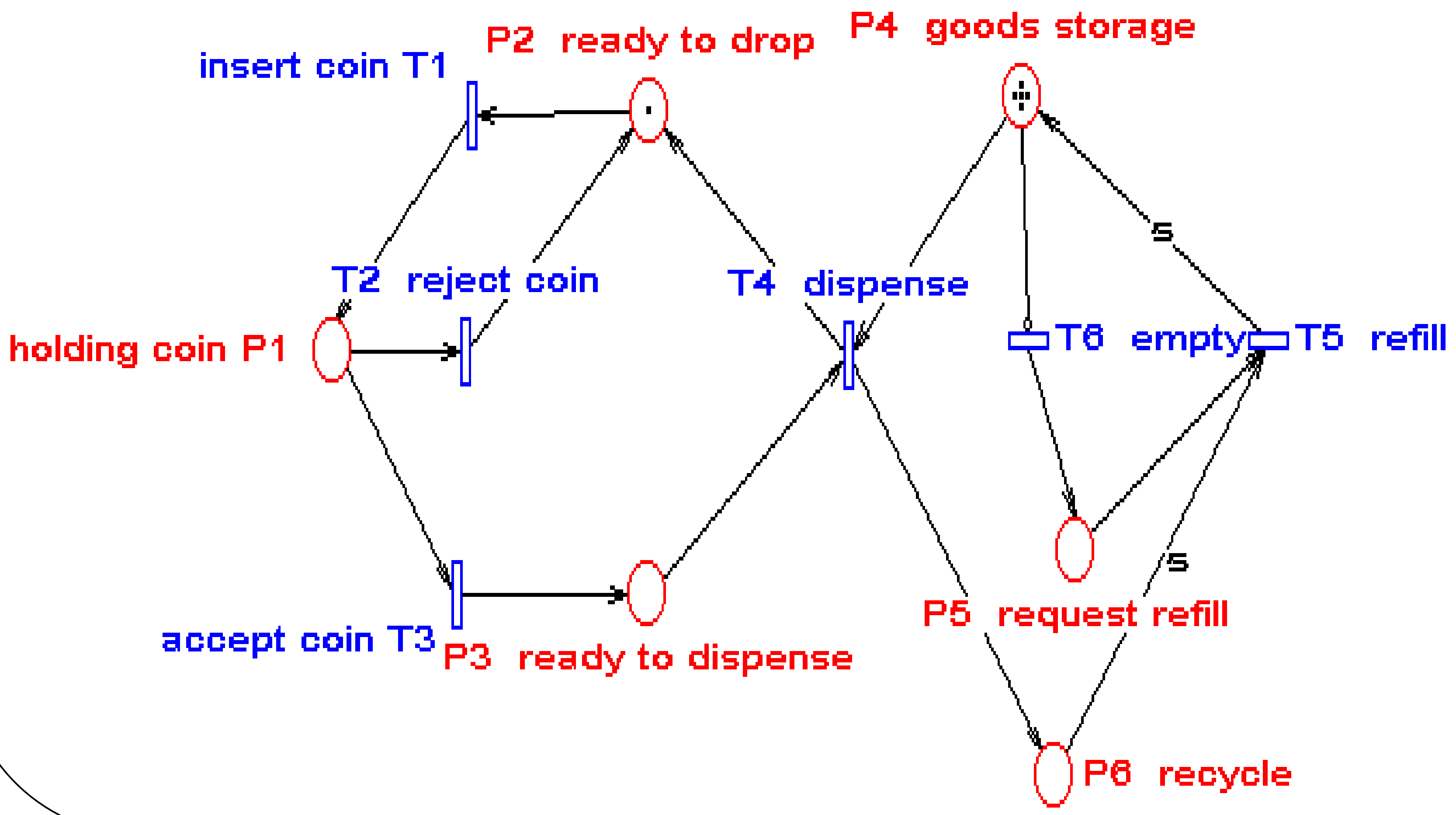
State is marking of places with tokens.

Intrinsic representation of state. Time must be introduced separately.

Modeling a Service Center as a Petri Net



This is a Petri net model of a vending machine stocked with 5 items.



Three Ways to Introduce Time Into Petri Nets

1. Firing Times - Associate a time delay with each transition. A transition once enabled, removes the tokens from its input places but does not fill the output arcs until the delay period has expired.

2. Holding Times - Tokens in output places have a delay associated with them. A token cannot be used to enable a transition until its delay period has expired. The delay period can be set by the creating transition or the holding place.

3. Enabling Times - A time delay is imposed between enabling and firing. A transition, once enabled, does not execute at all until the delay period has elapsed.

Abstract State Machines

Assume the execution behavior of a system can be specified as an interacting set of state machines exchanging events.

1. Abstract the functional behavior. Retain only the times taken to execute a function. – Performance models

2. Abstract the functional behavior. Retain only the times to execute a function. Add specifications for probability of failure of a component. – Reliability models.

3. Abstract the system to be a system with a finite number of states. – Functional correctness models.

State Machine Modeling Languages

1. Software-Oriented Languages

   StateCharts

   UML

2. Model Checking Languages

   Promela

   SMV

   S/R