Modeling Techniques

• Analytic
  – Queuing Networks
  – Stochastic Petri Nets
  – Generalized Stochastic Petri Nets
  – Markov Chains
  – Process Algebras

• Simulation

• Hybrid Models: combination of analytic and simulation models.
Petri Nets (PNs)

- Queuing Networks (QNs) are not good at expressing nor modeling concurrent events (e.g., fork and join situations, blocking, etc)
- Petri Nets (PNs) are good for representing concurrency but do not lend themselves to performance analysis.
- Adding time to PNs enable them to be used in modeling.

Review of Basic Petri Nets

- Directed bi-partite graph with two types of nodes:
  - Places: used to hold tokens.
  - Transitions
- Firing Rule: a transition \( t \) is enabled if all input arcs have at least one token. When a transition fires, a token is removed from each input place and one token is placed in each output place.
Review of PNs

Transition $t$ is not enabled.

Review of PNs

Transition $t$ is enabled.
Review of PNs

PN for Mutual Exclusion

© 2000 Daniel A. Menascé. All Rights Reserved.
PN for Mutual Exclusion

P1

CS

P3

NCS

CS

P2

CS

P4

NCS

PN for Mutual Exclusion

P1

CS

P3

NCS

CS

P2

CS

P4

NCS

© 2000 Daniel A. Menascé. All Rights Reserved.
Reachability Set

- A marking of a PN is a tuple of the form
  \[ M = (m_1, \ldots, m_p) \]
  where \( m_i \) is the number of tokens in place \( i \).
- \( R \): set of all possible markings
- \( M_0 \): initial marking.
Reachability Set for Mutual Exclusion Example

\[ (1, 1, 1, 0, 0) \]

\[ (1, 0, 1, 1, 0) \]

\[ (1, 1, 0, 0, 1) \]

Adding Time to PNs

- Transitions correspond to actions and places to conditions.
- Actions do not occur in zero time in real life.
- Make transitions take a time to fire. If firing time is exponentially distributed, then PN becomes a Stochastic Petri Net (SPN).
- If the SPN also allows instantaneous transitions as well as exponential transitions, then it is a Generalized Stochastic Petri Net (GSPN).
- Instantaneous transitions are used to specify control.
GSPN Graphical Notation

Instantaneous transition.

Timed transition.

Input places to timed transitions act like queues.
GSPN for Mutual Exclusion

M = (1, 1, 1, 0, 0)

© 2000 Daniel A. Menascé. All Rights Reserved.
GSPN for Mutual Exclusion

M = (1,0,1,1,0)

© 2000 Daniel A. Menascé. All Rights Reserved.

GSPN for Mutual Exclusion

M = (1,1,1,0,0)

© 2000 Daniel A. Menascé. All Rights Reserved.
GSPN for Mutual Exclusion

\[ M = (1,1,0,0,1) \]

© 2000 Daniel A. Menascé. All Rights Reserved.
GSPN for Mutual Exclusion

\[ M = (1,0,0,1,1) \]

GSPN for Mutual Exclusion

\[ M = (1,1,0,0,1) \]
GSPN for Mutual Exclusion

\[ M = (1,1,1,0,0) \]

Reachability Set for Mutual Exclusion Example

For SPNs, can build a Markov Chain from the reachability set.
Computing Performance Metrics from GSPNs

- Packages such as GSPN (http://www.di.unito.it/~greatspn/) allow one to:
  - Specify a GSPN using a graphic editor, language or combination.
  - Solve GSPN to obtain probabilities of each marking.
  - Simulate a GSPN.
  - Metrics of interest are associated with functions of marking probabilities.

Simulation

- Discrete event simulation:
  - Event generation
  - Calendar of events
  - Event processing procedures
  - Clock (simulated clock)
- Trace-drive simulation: part of the input data comes from traces of execution (e.g., memory references, HTTP logs, etc).
Components of a Simulation Model

- Event Generation:
  - Trace-driven
  - Distribution-driven
  - Hybrid
- Event Processing
  - Calendar of Events
  - Event-handling procedures
- Transaction List (with parameters)
- Queues
- Simulation Clock
- Computation of Statistics

Simulation Basics

- Simulation clock
- Get Next Event
- Event Processing Routine
- Update clock And Generate Events

<table>
<thead>
<tr>
<th>Time</th>
<th>EvType</th>
<th>Ev. Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Calendar of events
(sorted in chronological order)
Simulation Model Example:
Single Queue

• Events:
  – Arrival of a customer
  – Service completion

• Statistics:
  – Total number of arrivals
  – Total departures
  – Total server busy time
  – Total waiting time
  – Total departures from queue
  – Total squares of waiting time

Simulation Example

1. Generate new arrival event
2. Add arrival event to CE
3. Remove first event from CE and update clock

Event type?

- arrival
- service completion

4. Add request to queue
5. server idle?
  - yes
  - no

6. Remove request from queue
7. Generate svc. time
8. Add completion event to CE
9. Update Statistics
10. Remove first request from queue
11. Generate svc. time
12. Add completion event to CE
Calendar of Events

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event Time</th>
<th>Event Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival</td>
<td>10.5</td>
<td>.....</td>
</tr>
<tr>
<td>arrival</td>
<td>12.8</td>
<td>.....</td>
</tr>
<tr>
<td>completion</td>
<td>13.1</td>
<td>.....</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• The calendar of events is ordered in increasing chronological order.
• Parameters may include the transaction Id associated with the event.

Common Mistakes in Simulation

• Inappropriate level of detail:
  – Too detailed: more development time and higher likelihood of bugs
  – Should start with a less detailed model first and increase complexity as needed.

• Unverified Models:
  – Simulation programs are usually large and complex programs and may have bugs that invalidate the results.

• Invalid Models:
  – Incorrect assumptions may be used. Need to validate through analytic models, measurements, and or intuition.
Common Mistakes in Simulation

• Improperly Handled Initial Conditions:
  – Should discard first part of run: transient behavior.

![Graph showing time and discard](image)

Common Mistakes in Simulation

• Improper simulation length.
• Poor Random Number Generator.
• Improper Selection of Seeds.
Verifying Simulation Models

• Trace Analysis: examine traces of a few transactions as they go through the system.
• Continuity Test: small variations in the input should show small variations in the output.
• Check Extreme Values: extreme values (e.g., low loads or very high loads) should be easy to verify by crude analytic models.

Verifying Simulation Models

• Check for Basic Relationships: verify if results satisfy basic laws (e.g., Little’s Law).
• Bound validation: use, if possible, existing analytic models for situations that are known to be upper or lower bounds
• Trend verification: check if the trends shown by the model match your intuition.
• Numeric range validation: check if the numerical results are within expected numerical ranges.
Transient Elimination with Independent Runs

- Run $m$ runs of the simulation with a different seed for each run.
- Each run has $n$ observations.
- Let $x_{i,j}$ be the j-th observation in the i-th run.

\[
\begin{array}{cccccc}
\text{run 1} & 1 & 2 & 3 & \ldots & n \\
\text{run m} & 1 & 2 & 3 & \ldots & n \\
\end{array}
\]

Step 1: Compute average of j-th observation over all runs.

\[
\bar{x}_j = \frac{1}{m} \sum_{i=1}^{m} x_{i,j}
\]

Step 2: Compute the overall average.

\[
\bar{x} = \frac{1}{n} \sum_{j=1}^{n} \bar{x}_j
\]

Step 3: Set the number of deleted observation, $k$, equal to 1.

Step 4: Compute the overall mean without the first $k$ observations.

\[
\bar{x}_k = \frac{1}{n-k} \sum_{j=k+1}^{n} \bar{x}_j
\]
Transient Elimination with Independent Runs

Step 5: compute the relative change $\Delta$

$$\Delta = \frac{\bar{x}_k - \bar{x}}{\bar{x}}$$

Step 6: If $|\Delta| > \text{tolerance}$ then do $k ← k + 1$ and go to step 4.

Step 7: Remove the first $k$ observations and use $\bar{x}_k$ as the average.

Transient Elimination with Batch Means

- Single run with $N$ observations.
- Divide the run into $m$ sub-samples called batches of size $n = \lfloor N / m \rfloor$.
- Let $x_{i,j}$ be the $j$-th observation in the $i$-th batch.
Transient Elimination with Batch Means

Step 1: Set \( n = 10 \).

Step 2: compute the average of the \( i \)-th batch.

\[
\bar{x}_i = \frac{1}{n} \sum_{j=1}^{n} x_{i,j}
\]

Step 3: compute the overall average.

\[
\bar{x} = \frac{1}{m} \sum_{i=1}^{m} \bar{x}_i
\]

Step 4: Compute the variance of the batch means:

\[
Var(\bar{x}) = \frac{1}{m-1} \sum_{i=1}^{m} (\bar{x}_i - \bar{x})^2
\]

Step 5: Increase \( n \) by 10 and repeat steps 2-4 and plot the variance as a function of \( n \). The point at which the variance starts to decreases is the length of the transient interval.
Stopping Criteria
Independent Runs

- Run $m$ runs of the simulation with a different seed for each run.
- Each run has $n_n + n_o$ observations where $n_n$ is the size of the transient phase.
- The number $n$ is increased until the precision in the confidence interval reaches a desired value.

Step 0: Initialization: $n = 100$.

Step 1: compute the mean for each replication.

$$\bar{x}_i = \frac{1}{n} \sum_{j=n_n+1}^{n} x_{i,j}$$
Stopping Criteria
Independent Runs

Step 2: compute the overall mean for all replications.

$$\bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

Step 3: compute the variance of the replicate means.

$$\text{Var}(\bar{x}) = \frac{1}{m-1} \sum_{i=1}^{m} (\bar{x}_i - \bar{x})^2$$

Step 4: compute the confidence interval for the mean as:

$$\bar{x} \pm t_{[1-\alpha/2,m-1]} \frac{\sqrt{\text{Var}(\bar{x})}}{\sqrt{m}}$$

Step 5: compute the accuracy $r$ as.

$$r = \left( \frac{t_{[1-\alpha/2,m-1]} \sqrt{\text{Var}(\bar{x})}}{\sqrt{m} \bar{x}} \right) \times 100$$

Step 6: If $r >$ desired value (e.g., 5) then $n = n + 100$ and go to Step 1, else STOP.
Stopping Criteria
Independent Runs

- Number of discarded observations: $m \times n_o$
- To reduce the number of wasted observations use a small value of $m$.

Stopping Criteria
Batch Means

- Single run with $N+n_o$ observations where $n_o$ is the size of the transient phase.

Step 0: Start with a small value of $n$ (e.g., 1).
Step 1: compute the mean for each batch.

$$\bar{x}_i = \frac{1}{n} \sum_{j=1}^{n} x_{i,j}$$
Stopping Criteria
Batch Means

Step 2: compute the overall mean for all batches.

\[ \bar{x} = \frac{1}{m} \sum_{i=1}^{m} \bar{X}_i \]

Step 3: compute the variance of the batch means.

\[ \text{Var}(\bar{x}) = \frac{1}{m-1} \sum_{i=1}^{m} (\bar{X}_i - \bar{x})^2 \]

Step 4: compute the confidence interval for the mean as:

\[ \bar{x} \pm t_{[1-\alpha/2;m]} \sqrt{\frac{\text{Var}(\bar{x})}{m}} \]

Step 5: compute the auto-covariance

\[ \text{Cov}(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2} \sum_{i=1}^{m-1} (\bar{X}_i - \bar{x})(\bar{X}_{i+1} - \bar{x}) \]

Step 6: Check for proper batch size: If \( \text{Cov}(\bar{x}_i, \bar{x}_{i+1}) \ll \text{Var}(\bar{x}) \) then stop. Otherwise, double \( n \) and go to step 1.
Seed Selection

- Never use zero as a seed.
- Avoid even values.
- Reuse seed for repeatability of experiments.
- Do not use random seeds (e.g., system time) if the simulation is to be repeated.

Generation of Random Variables

- Assume that \( u \) is a value uniformly distributed between 0 and 1.
- Method of the inverse of the CDF:
Generation of Random Variables

- Assume that $u$ is a value uniformly distributed between 0 and 1.
- CDF for the exponential: $1 - e^{-\frac{x}{\alpha}}$
  - Inverse of the CDF: $-\alpha \ln(u)$
- CDF for the Pareto distribution: $1 - x^{-a}$
  - Inverse of the CDF: $\frac{1}{u^{\frac{1}{\alpha}}}$

Simulation Programs

- Written in general purpose programming languages (e.g., C, C++).
- Written in high-level programming languages with the help of simulation libraries (e.g., SMPL, Simpack, CSim).
- Using special purpose simulation languages (e.g., GPSS/H).
- Using simulation packages (e.g., SES Workbench, OPNET).