Co-funded by the
Erasmus+ Programme
of the European Union

DeCAIR

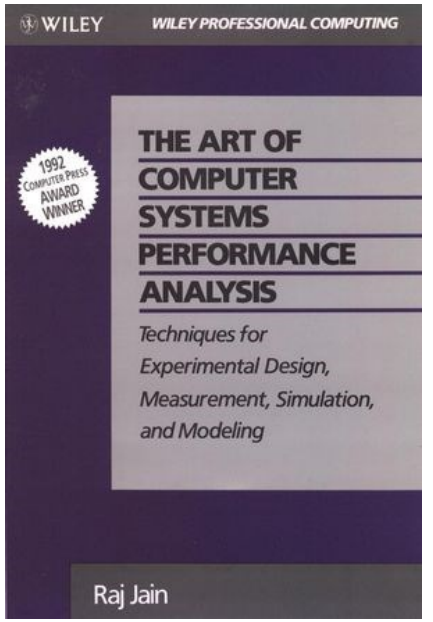
Simulation

Prof. Gheith Abandah

Developing Curricula for Artificial Intelligence and Robotics (DeCAIR)
618535-EPP-1-2020-1-JO-EPPKA2-CBHE-JP

References

- Raj Jain, **The Art of Computer Systems Performance Analysis**, Wiley, 1991.
  - Part I: An Overview of Performance Evaluation
  - Part II: Measurement Techniques and Tools
  - Part III: Probability Theory and Statistics
  - Part IV: Experimental Design and Analysis
  - Part V: Simulation



Outline

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Common Mistakes in Simulation

1. **Inappropriate Level of Detail**
   More detail $\Rightarrow$ More time $\Rightarrow$ More Bugs $\Rightarrow$ More CPU
   $\Rightarrow$ More parameters $\neq$ More accurate
2. **Improper Language**

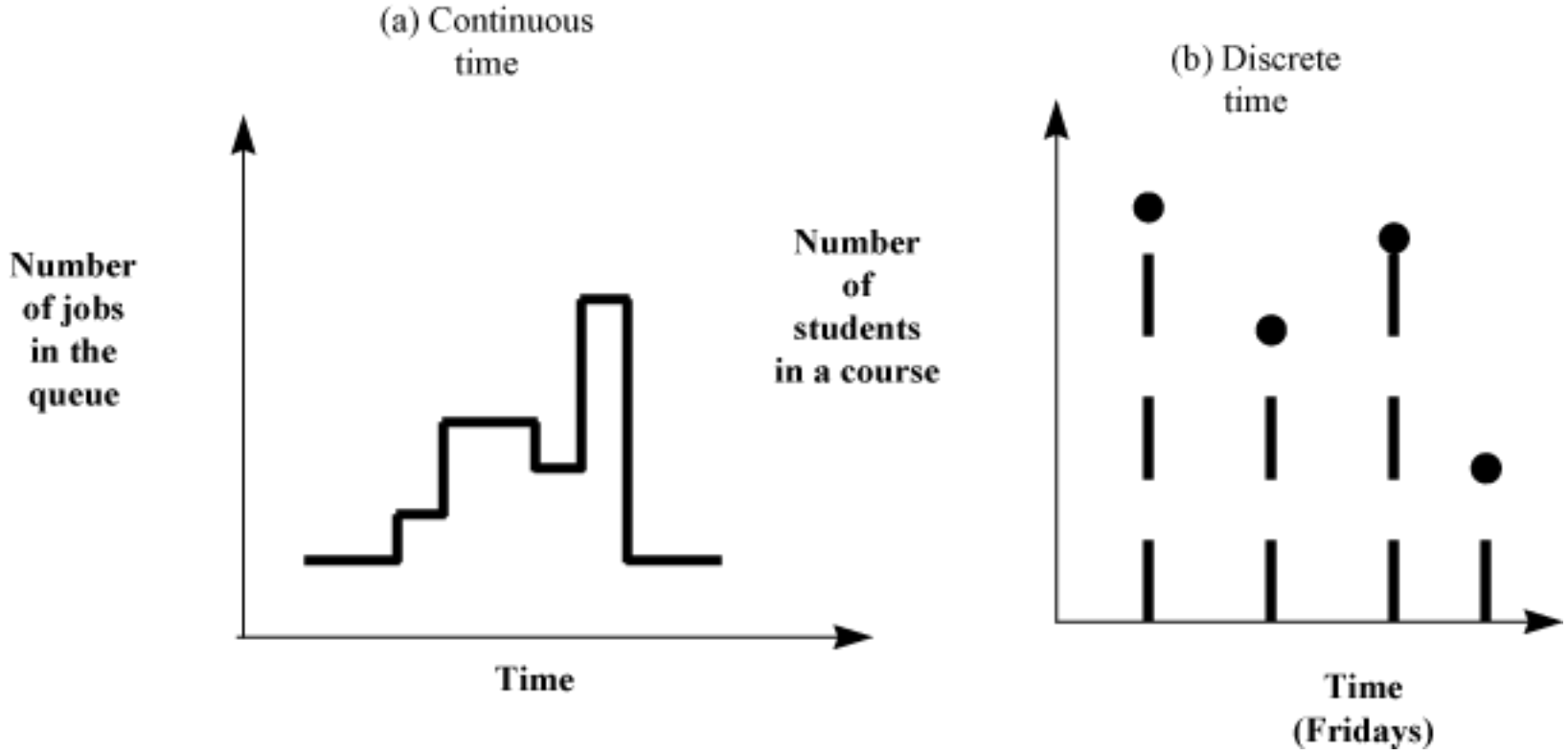
   General purpose $\Rightarrow$ More portable, More efficient, More time
3. **Unverified Models**: Bugs
4. **Invalid Models**: Model *vs*. reality
5. **Improperly Handled Initial Conditions**
6. **Too Short Simulations**: May be dominated by the transient part

Outline

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

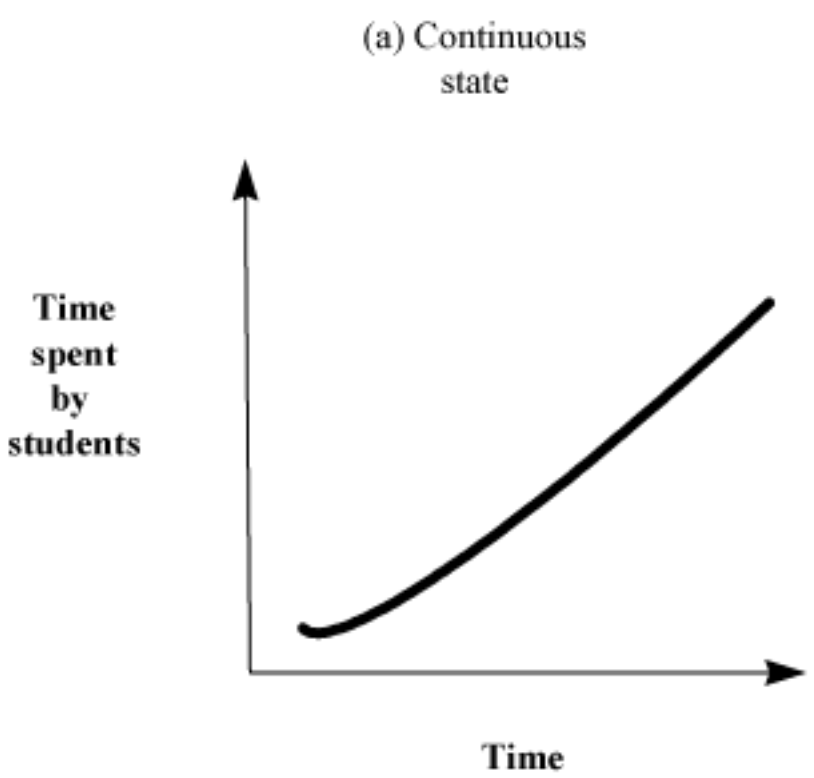
Types of Models

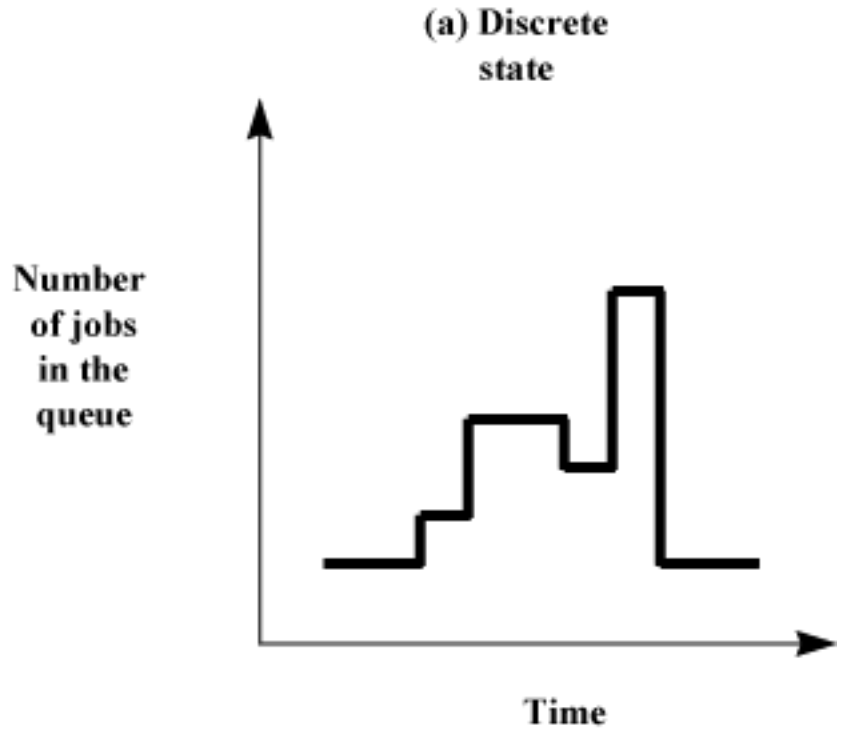
- **Continuous Time Models**: State is defined at all times.
- **Discrete Time Models**: State is defined only at some instants.



Types of Models (cont.)

- **Continuous State Models**: State variables are continuous.
- **Discrete State Models**: State variables are discrete.





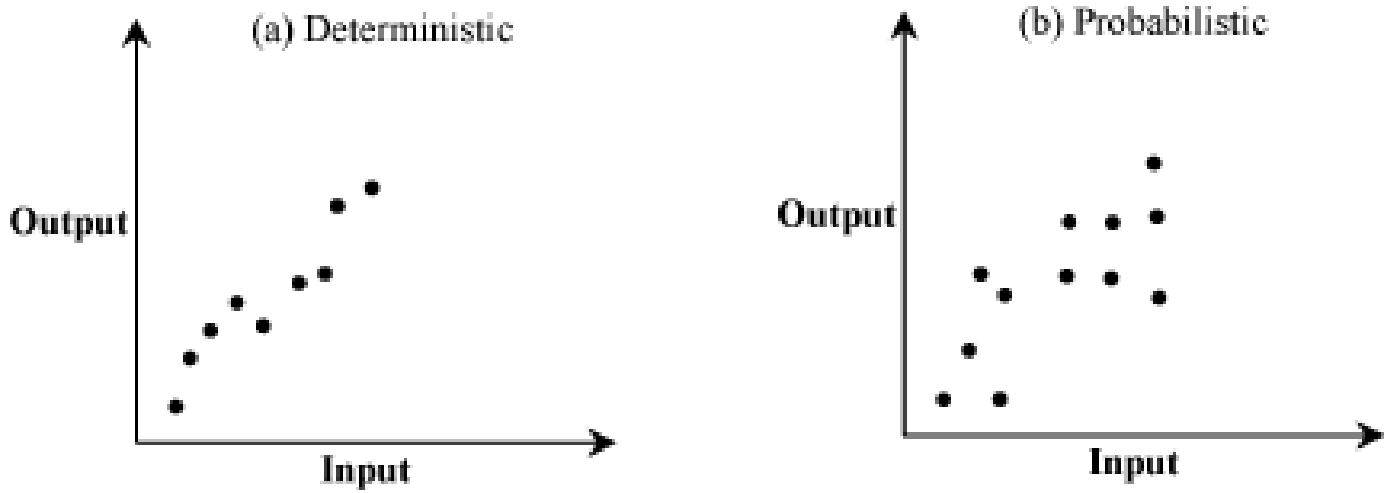
Types of Models (cont.)

- Discrete state = Discrete event model
- Continuous state = Continuous event model
- Continuity of time ≠ Continuity of state

- **Four possible combinations**
  1. Discrete state/discrete time models
  2. Discrete state/continuous time models
  3. Continuous state/discrete time models
  4. Continuous state/continuous time models

Types of Models (cont.)

- **Deterministic and Probabilistic Models**

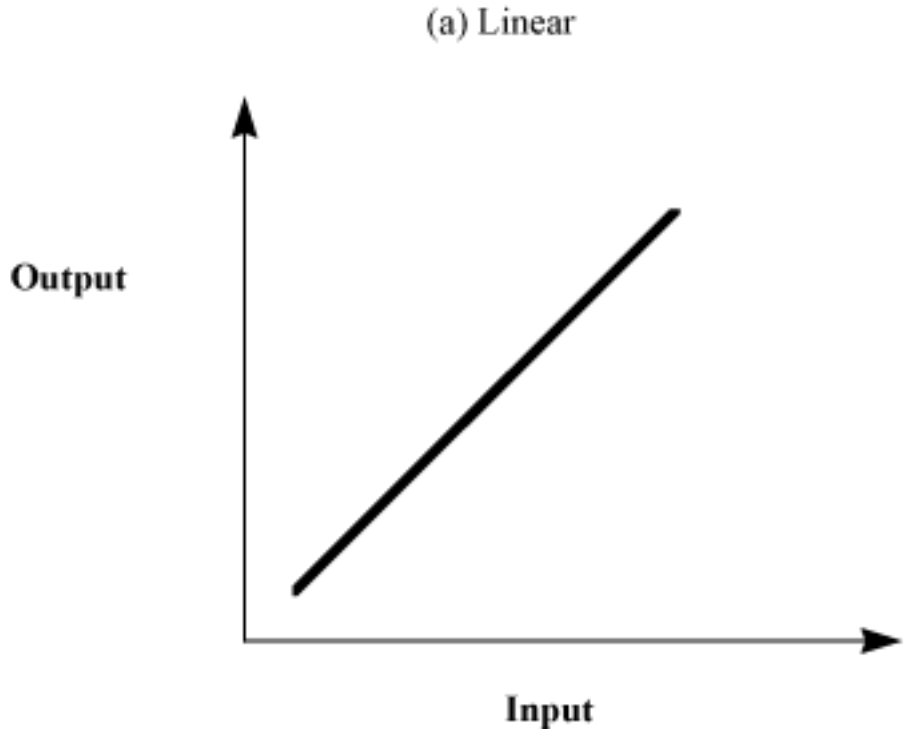


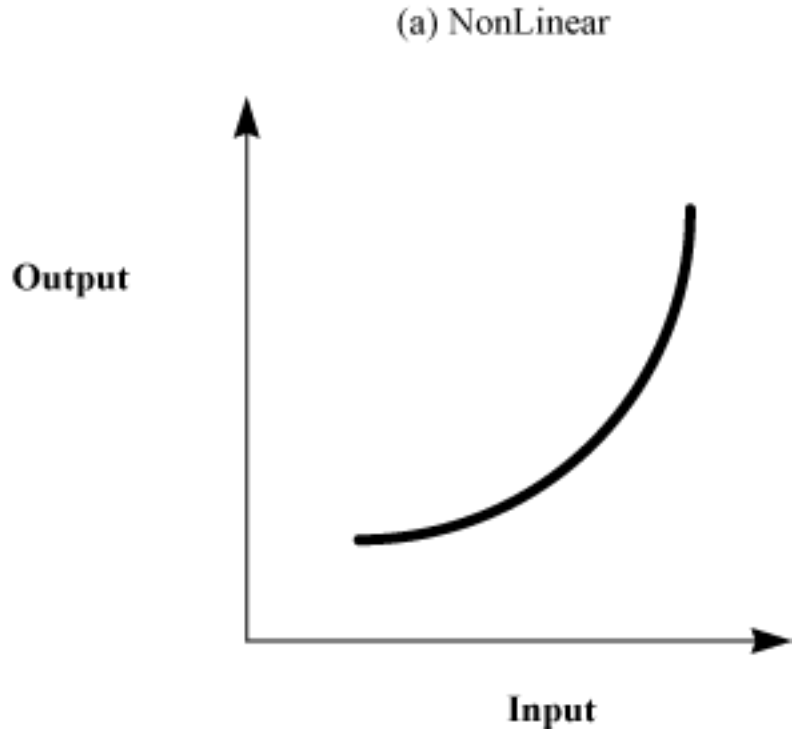
- **Static and Dynamic Models**

  CPU scheduling model *vs*. $E = mc^2$.

Types of Models (cont.)

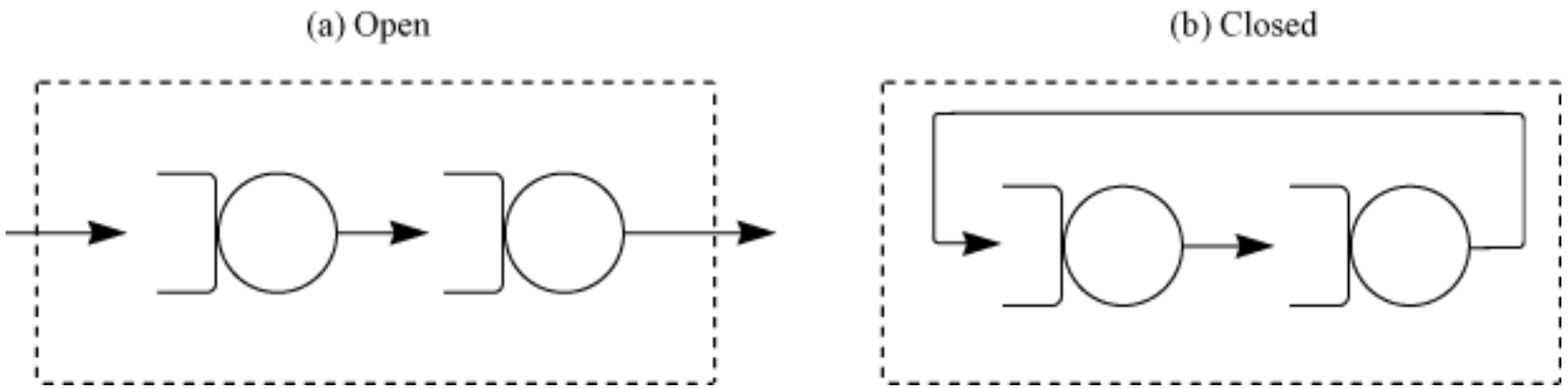
- **Linear and Nonlinear Models**: Depending on the input/output relation.





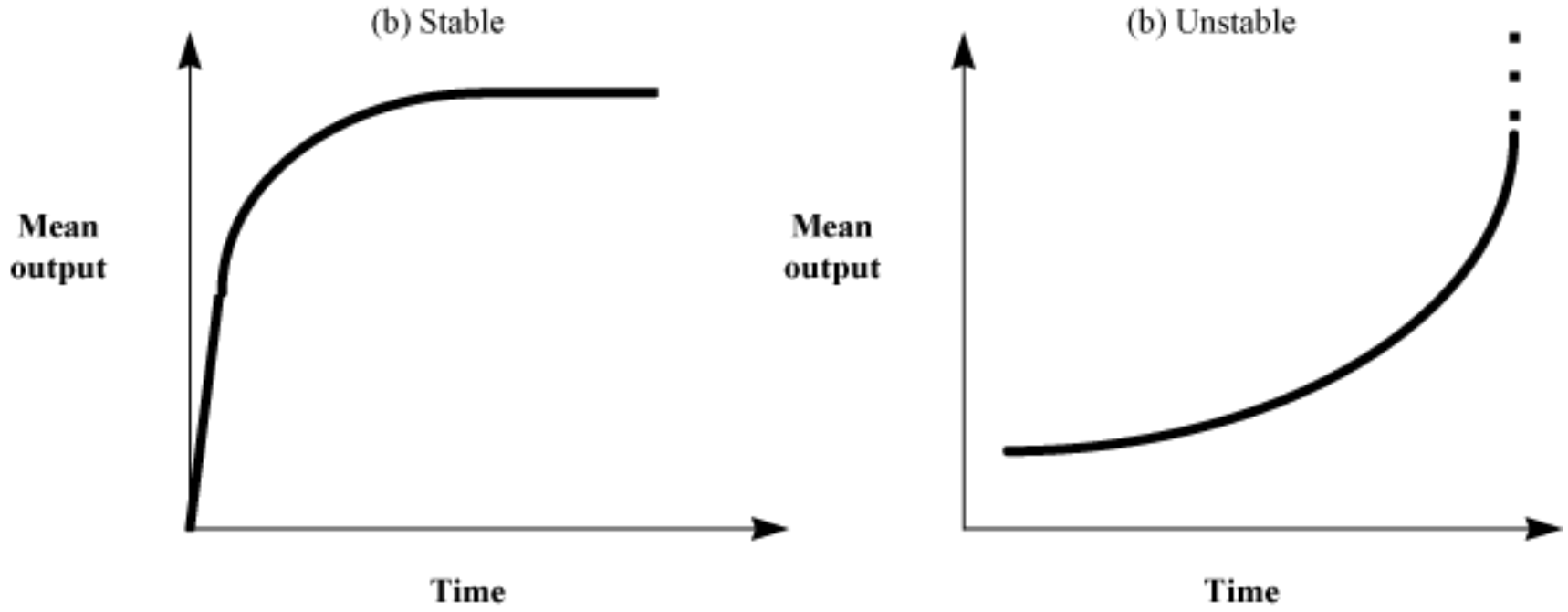
Types of Models (cont.)

- **Open and Closed Models**: It is open when the input is external.



Types of Models (cont.)

- **Stable and Unstable Models**
  - Stable → Settles to a steady state
  - Unstable → Continuously changing



Computer System Models

- Continuous time
- Discrete state
- Probabilistic
- Dynamic
- Nonlinear
- Open or closed
- Stable or unstable

Outline

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Selecting a Language for Simulation

1. General purpose language
2. Extension of a general-purpose language
3. Simulation language
4. Simulation package

1. General Purpose Language

- **Advantages**
  - Analyst's familiarity
  - Easy availability
  - Quick startup
  - Efficiency, flexibility, and portability
- **Disadvantage**: Need time to develop routines for event handling, random number generation, *etc*.
- **Examples**: Python, C++

2. Extension of a General-Purpose Language

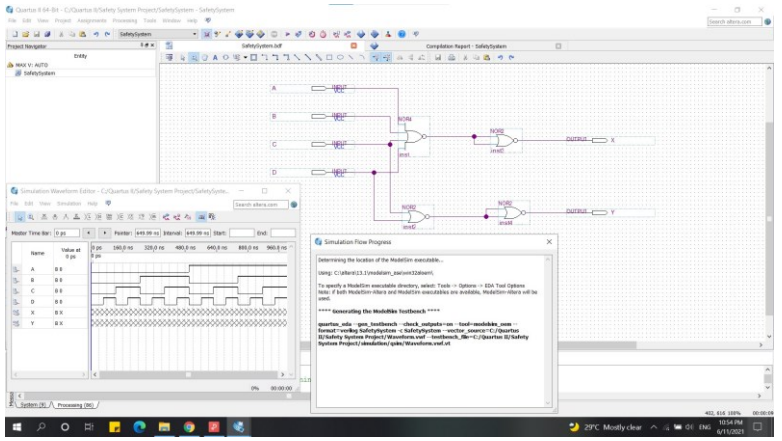
- Collection of routines to handle common simulation tasks.
- Compromise for efficiency, flexibility, and portability.
- **Examples**: SimPy for Python and JSL for Java.

3. Simulation Language

- Saves development time
- Has built-in facilities for time advancing, event scheduling, entity manipulation, random variate generation, statistical data collection, and report generation.
- More time for system specific issues.
- Very readable modular code.
- **Examples**: Verilog, VHDL

4. Simulation Package

- Define the model using a wizard dialog or drawing canvas.
- Big time savings
- Inflexible ⇒ Simplification
- **Examples**: Quartus, NS2, SimpleScalar



Outline

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Important Simulation Types

1. **Emulation**: Using hardware or firmware

   E.g., Terminal emulator, processor emulator

   Mostly hardware design issues
2. **Monte Carlo Simulation**
3. **Trace-Driven Simulation**
4. **Discrete Event Simulation**

2. Monte Carlo Simulation

- Static simulation (no time axis)
- To model probabilistic phenomenon
- Need pseudorandom numbers
- Example: Approximating an integral

$$I = \int_0^2 e^{-x^2} dx$$

3. Trace-Driven Simulation

- Trace = Time ordered record of events on a system
- Trace-driven simulation = Trace input
- The trace should be independent of the system under study
- **Example**: Trace of pages fetched depending upon the working set size and page replacement policy
  - Not good for studying other page replacement policies
  - Better to use pages referenced only

3. Trace-Driven Simulation (cont.)

- **Advantages**
  1. **Credibility**
  2. **Easy Validation**: Compare simulation with measured
  3. **Accurate Workload**: Models correlation and interference
  4. **Detailed Trade-Offs**: Detailed workload ⇒ Can study small changes in algorithms
  5. **Less Randomness**: Trace ⇒ deterministic input ⇒ fewer repetitions
  6. **Fair Comparison**: Better than random input
  7. **Similarity to the Actual Implementation**: Trace-driven model is similar to the system

3. Trace-Driven Simulation (cont.)

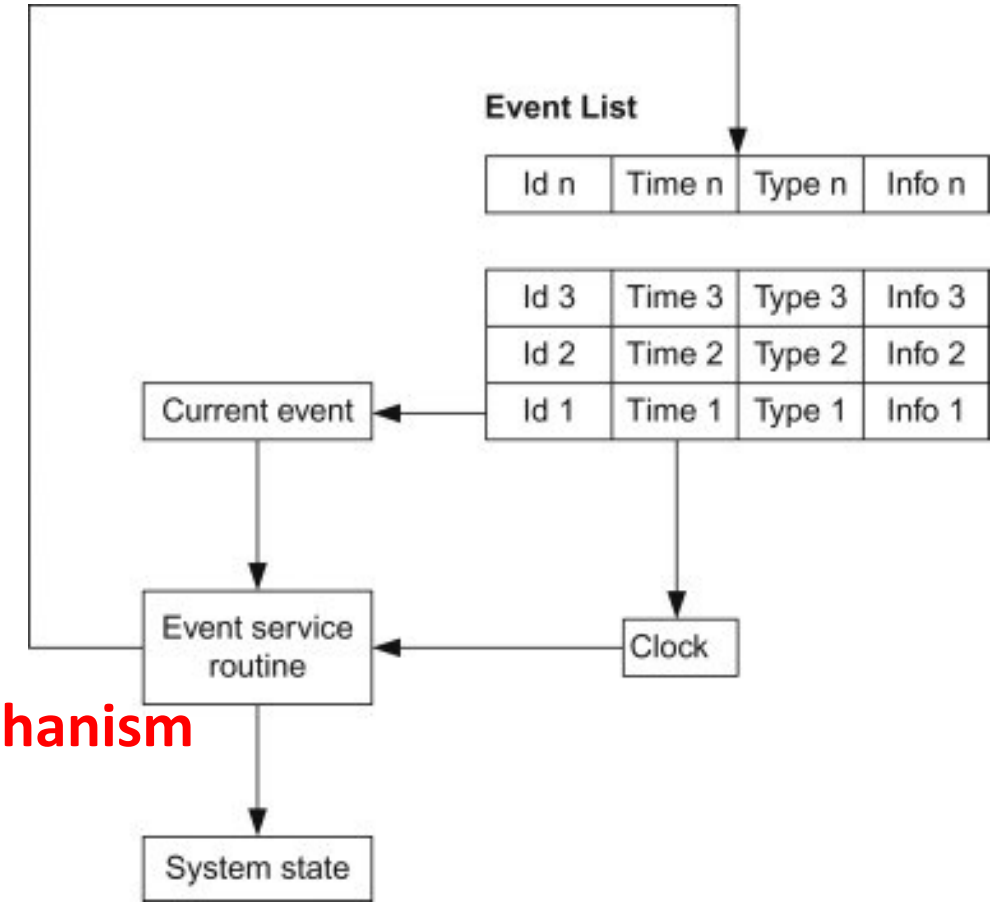
- **Disadvantages**
  1. **Complexity**: More detailed
  2. **Representativeness**: Workload changes with time
  3. **Finiteness**: Few minutes fill up a disk
  4. **Single Point of Validation**: One trace = one point
  5. **High Detail is Needed**
  6. **Trade-Off**: Difficult to change workload

4. Discrete Event Simulation

- Concentration of a chemical substance
  $\Rightarrow$ Continuous event simulations
- Number of jobs $\Rightarrow$ Discrete event
- Discrete state $\neq$ discrete time

4. Discrete Event Simulation (cont.)

- **Components**
  1. **Event Scheduler**
     (a) Schedule event X at time T.
     (b) Hold event X for a time interval dt.
     (c) Cancel a previously scheduled event X.
     (d) Hold event X indefinitely
     (e) Schedule an indefinitely held event.
  2. **Simulation Clock** and a **Time Advancing Mechanism**
     (a) Unit-time approach
     (b) Event-driven approach



Outline

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Model Verification *vs*. Validation

- Verification $\Rightarrow$ Debugging
- Validation $\Rightarrow$ Model = Real world

- Four Possibilities:
  1. Unverified, Invalid
  2. Unverified, Valid
  3. Verified, Invalid
  4. Verified, Valid

Model Verification Techniques

1. Top-Down Modular Design
2. Anti-bugging
3. Structured Walk-Through
4. Deterministic Models
5. Run Simplified Cases
6. Use Traces
7. On-Line Graphic Displays
8. Continuity Test
9. Degeneracy Tests
10. Consistency Tests
11. Seed Independence

Model Verification Techniques (cont.)

1. Top-Down Modular Design

- Divide and Conquer
- Modules = subroutines, subprograms, procedures
  - Modules have well-defined interfaces
  - Can be independently developed, debugged, and maintained
- Top-down design
  - Hierarchical structure
  - Modules and sub-modules

Model Verification Techniques (cont.)

2. Anti-bugging: Include self-checks:

$\sum$ Probabilities = 1

Jobs left = Generated - Serviced

3. Structured Walk-Through

- Explain the code to another person or group
- Works even if the person is sleeping

4. Deterministic Models: Use constant values

5. Run Simplified Cases

- Only one packet
- Only one source
- Only one intermediate node

Model Verification Techniques (cont.)

6. Use Traces

- Several levels of detail:
  - Events trace
  - Procedures trace
  - Variables trace
- User selects the detail
  - Include on and off

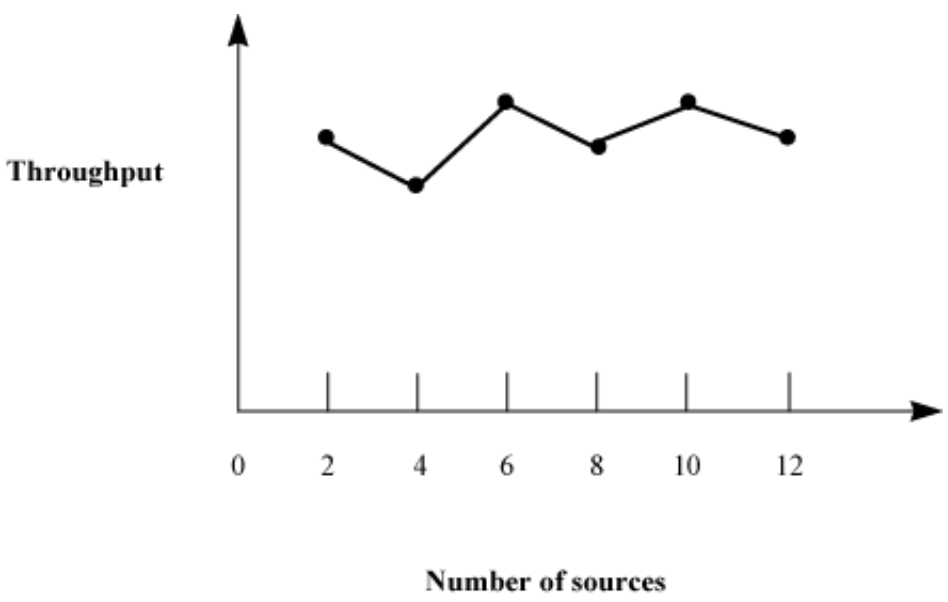
7. On-Line Graphic Displays

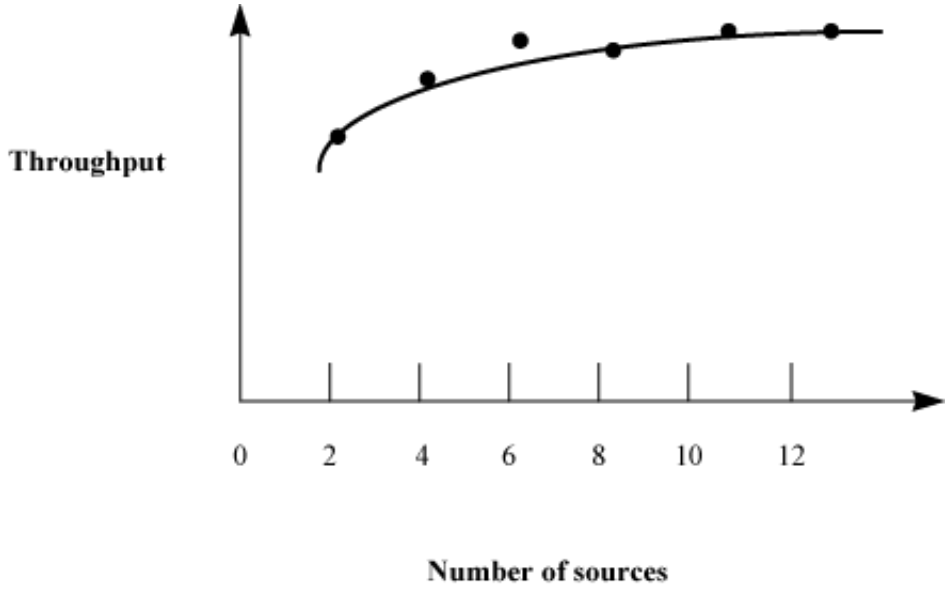
- Make simulation interesting
- Help selling the results
- More comprehensive than trace

Model Verification Techniques (cont.)

8. Continuity Test

- Run for different values of the input parameters
- Slight change in input $\Rightarrow$ slight change in output
- Example Problem:





Model Verification Techniques (cont.)

9. Degeneracy Tests: Try extreme configuration and workload

- One CPU, Zero disk

10. Consistency Tests

- Similar result for inputs that have same effect, *e.g.*, 4 users at 100 Mbps *vs*. 2 at 200 Mbps

11. Seed Independence: Similar results for different seeds

Outline

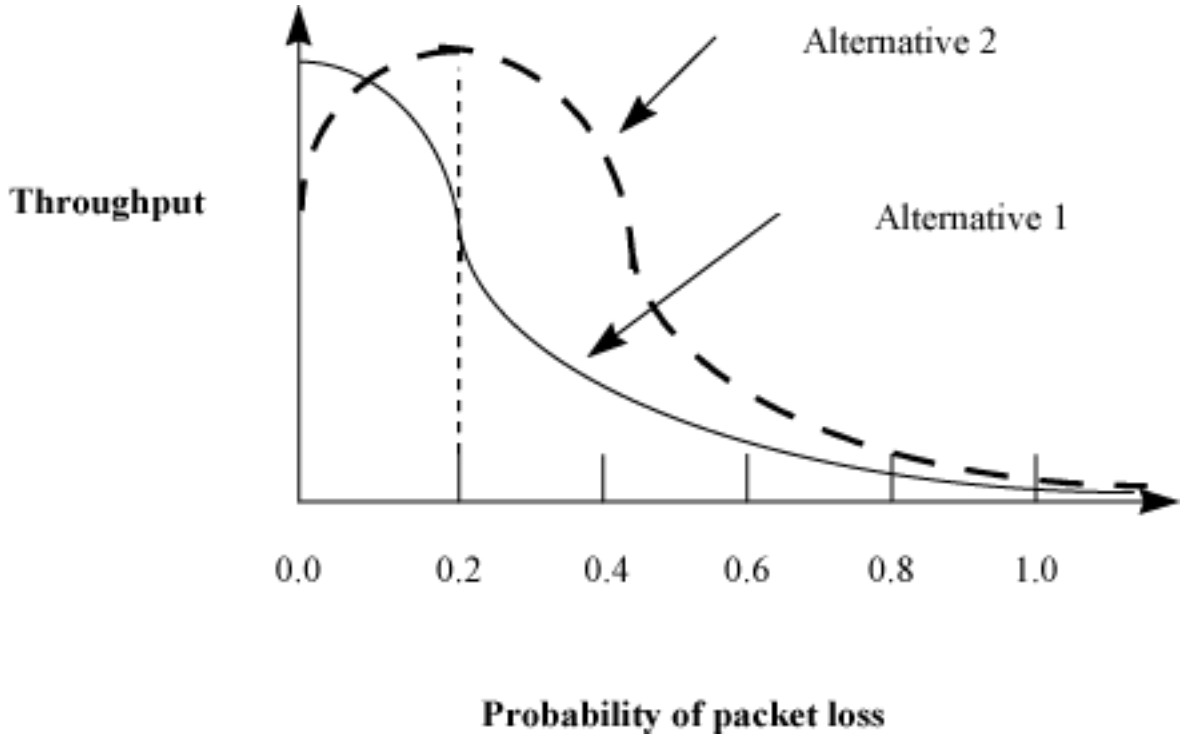
- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Model Validation Techniques

- Validation techniques for one problem may not apply to another problem.
- **Aspects to Validate**
  1. Assumptions
  2. Input parameter values and distributions
  3. Output values and conclusions
- **Techniques**
  1. Expert intuition
  2. Real system measurements
  3. Theoretical results

1. Expert Intuition

- Most practical and common way
- **Experts** = Involved in design, architecture, implementation, analysis, marketing, or maintenance of the system
- **Expert selection** = function of Life cycle stage
- Present assumption, input, output
- Better to validate one at a time
- See if the experts can distinguish simulation *vs*. measurement.



2. Real System Measurements

- Compare assumptions, input, and output with the real world.
- Often infeasible or expensive.
- Even one or two measurements add to the validity.

3. Theoretical Results

- Analysis = Simulation
- Used to validate analysis also
- Both may be invalid
- Use theory in conjunction with experts' intuition
  - E.g., Use theory for a large configuration
  - Can show that the model is not invalid

Outline

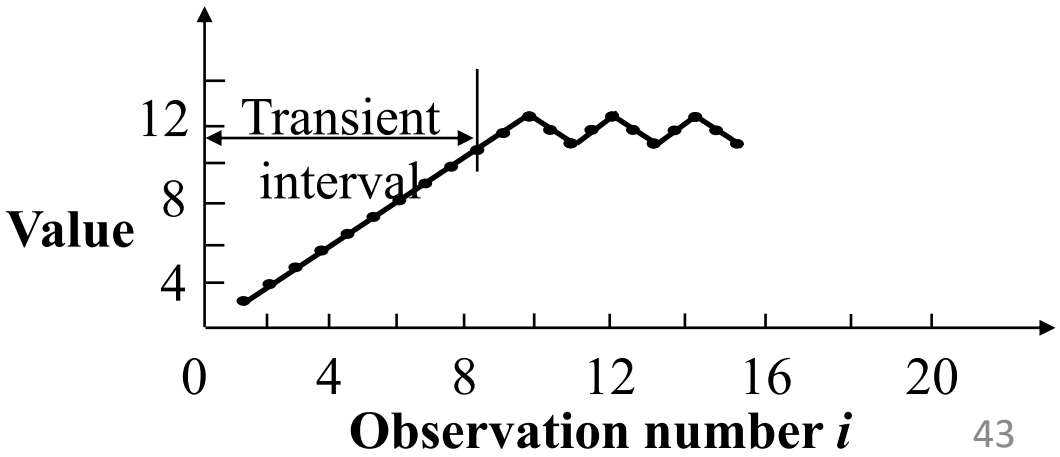
- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal

Transient Removal

- Generally steady state performance is interesting
- May need to remove the initial transient part

Transient Removal Techniques

1. **Long Runs**
   - Wastes resources
   - Difficult to ensure that it is long enough
2. **Proper Initialization**
   - Start in a state close to the expected steady state
     $\Rightarrow$ Reduces the length and the effect of the transient state
3. **Initial Data Deletion**
   - Delete some initial observation



Summary

- Common Mistakes in Simulation
- Types of Models
- Selecting a Language for Simulation
- Important Simulation Types
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal