Introduction to Simulation

Dr. John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@cs.rice.edu

Topics for Today

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Selecting a simulation language
- Types of simulations
- Scheduling events

More Simulation Topics

- **Next class**
  - —how to verify and validate a simulation
  - —how long should you run a simulation?
  - —how can the same accuracy be achieved with shorter run?
- **Next week**
  - —random number generation
  - —how to select RNG seeds
  - —how to verify that a RNG is good
  - —how to generate random variables with a given distribution

The Role of Simulation

- **Why simulation?**
  - —system under study may not be available
    - – common in design and procurement stages
  - —simulation may be preferred alternative to measurement
    - – controlled study of wider range of workloads and environments
  - —higher accuracy results than analytical modeling
- **Why not?**
  - —accurate simulation models take a long time to develop
    - – typically the evaluation strategy that takes the longest

Review of Evaluation Techniques

Criterion	Analytical Modeling	Simulation	Measurement
Stage	any	any	post-prototype
Time required	small	medium	varies
Tools	analysts	programs	instrumentation
Accuracy	low	moderate	varies
Trade-off evaluation	easy	moderate	difficult
Cost	low	medium	high
Saleability	low	medium	high

Evaluation Rules of Thumb

- **Combining evaluation techniques is useful**
  - —analytical model: find interesting range of parameters
  - —simulation: study performance within parameter range
- **Until validated, all evaluation results are suspect!**
  - —always validate one analysis modality with another
  - —beware of counterintuitive results!

Common Mistakes in Simulation

Common Mistakes: Too Much Detail

Inappropriate level of detail

- Level of detail limited only by time available for development
- A detailed model may not be a better model
  - —may require more detailed knowledge of input parameters
    - – inaccurate assumptions can yield wrong results
    - – example: time to service disk requests for timesharing simulation
      - could use exponential distribution for time for request service
      - could simulate disk rotation and head movement
      - but simulation better only if sector & track locations known
  - —may take too much time to develop
- Recipe for success
  - —start with less-detailed model
  - —get some results
  - —study sensitivities
  - —introduce details in key areas that affect results most

Common Mistakes: Programming Language

- Programming language = major impact on development time
- Special-purpose languages
  - —examples
    - – Facile [Larus Hill, Schnarr PLDI 2001]
      - language and compiler for processor simulators
  - —require less model development
  - —simplify several common tasks, e.g.
    - – verification using traces
    - – statistical analysis
    - – “fast-forwarding” of simulations
- General-purpose languages
  - —more portable
  - —provide more control over efficiency and run-time
  - —lack support for model development

Common Mistakes: Unverified Models

- Simulations are computer programs
- Bugs and programming errors are common
- Need to verify models to avoid wrong conclusions
  - —check that the model does what it is intended to do
    - – check whether simulation implements assumptions properly

Common Mistakes: Invalid Models

- Even if simulation has no errors it may not be representative
  - —assumptions: must validate representativeness
    - – otherwise, simulated behavior will not be representative
- All simulation results are suspect
- Must confirm with at least one of
  - —analytical model
  - —measurements
  - —intuition

Common Mistakes: Initial Conditions

Improper handling of initial conditions

- **Initial part of a simulation is generally not representative**
  - —transient behavior rather than steady state
- **Initial part of simulation should be discarded**
  - —several techniques for identifying beginning of steady state

Common Mistakes: Too Short Simulations

- Simulation run times are often very long
- Temptation is to halt simulations ASAP
- However
  - —results may be heavily dependent on initial conditions
  - —may not be representative of a real system until steady state
- Correct length for simulations depends on
  - —accuracy desired (width of confidence intervals)
  - —variance of observed quantities

Common Mistakes: Bad Random Numbers

- Bad random numbers can pollute simulation results
- How can random numbers be bad?
  - period too short
  - assume global randomness = local randomness
  - rely on bit subsets: may not be as random as whole
- Rule of thumb
  - use well-known generator rather than rolling your own
- Even well-known generators have had problems
- Improper selection of RNG seeds
  - seeds for different random streams must be carefully chosen
    - must ensure independence of streams
    - sources of error
      - share one stream for several different processes
      - use same seed for all streams
  - impact: introduce correlation among processes that may lead to non-representative results

Causes of Simulation Analysis Failure I

- Inadequate time estimate: underestimate effort required
  - —often start off as 1-week or 1-month projects
  - —continue for years
    - good: more features, parameters to provide better detail
    - bad: add more detail in hope of making it useful
- No achievable goal
  - —should have SMART goals
    - specific, measurable, achievable, repeatable, thorough
  - —not measurable: to model X
  - —projects without goals continue until funding runs out
- Incomplete mix of essential skills for a simulation project
  - —project leadership: lead, motivate, manage
  - —modeling and statistics: identify and model key characteristics at required level of detail
  - —programming: construct readable and verifiable program
  - —knowledge of modeled system: understand model, interpret results and their implications

Causes of Simulation Analysis Failure II

- **Inadequate level of user participation**
  - —modeling team and users must discuss system changes
    - – most systems change
    - – models developed in a vacuum rarely succeed
- **Obsolete or nonexistent documentation**
  - —most simulation models evolve over time as system does
  - —if system documentation is obsolete, modeling errors are likely
  - —best to use literate programming to keep documentation in sync
- **Inability to manage development of large, complex programs**
  - —SWE tools can help track
    - – design objectives
    - – functional requirements
    - – data structures
    - – progress estimates
  - —other useful principles
    - – top-down design
    - – structured programming
  - —without tools and techniques, hard to develop large models successfully

Causes of Simulation Analysis Failure III

Mysterious results

- **Causes**
  - —bugs in simulation program
  - —invalid modeling assumptions
  - —lack of understanding of system to be modeled
- **What to do?**
  - —attempt to verify the model
  - —bring persistent mysterious results to attention of users
    - may lead to unexpected insight into system
    - may point to system features that must be modeled in more detail

Simulation Checklist: Before Development

- Is the goal of the simulation properly specified?
- Is the level of detail in the model appropriate for the goal?
- Does the team include appropriate personnel?
  - —leadership, statistics and modeling, programming, and system
- Has sufficient time been allotted for the project?

Simulation Checklist: During Development

- Has the random number generator been tested?
  - —uniformity
  - —independence
- Is the model reviewed regularly with the end user?
- Is the model documented?

Simulation Checklist: During Execution

- **Is the simulation length appropriate?**
- **Are initial transients removed before computation?**
- **Has the model been verified thoroughly?**
- **Has the model been validated before using its results?**
- **If there are any surprising results, have they been validated?**
- **Are all seeds such that random streams will not overlap?**

Simulation Terminology

- State variables: variables that define the state of system
- Event: change in system state
- Continuous-time vs. discrete-time models
  - —continuous-time model: system state is defined at all times
  - —discrete-time model: state defined only at particular instants
- Continuous-state vs. discrete-state models
  - —classified by type of variables: continuous or discrete
    - continuous: uncountably infinite values
    - discrete: countable
  - —AKA continuous-event and discrete event models
- Deterministic vs. probabilistic models
  - —deterministic: output can be predicted with certainty
  - —probabilistic: sometimes a different result for same input parameters

More Simulation Terminology

- **Static vs. dynamic models**
  - —static: time is not a model variable
- **Linear vs. non-linear models**
- **Open vs. closed models**
  - —open: input from outside the model
    - – queuing model with arcs from outside
  - —closed: no external input
- **Stable vs. unstable models**
  - —stable: behavior settles down to steady state that is independent of time
  - —unstable: continuously changing behavior

Selecting a Language for a Simulation

Four choices

- **Simulation language**
  - —e.g. SIMSCRIPT (http://www.simprocess.com/products/simscript_description.cfm)
    - – built-in support for

      advancing time, scheduling events, manipulating entities, generating random variates, collecting statistics, generating reports, graphical representations & animation, checkpoint/restart, debugging
- **General-purpose language: chosen for language familiarity**
  - —must invest time developing core utilities
  - —other considerations: efficiency, flexibility and portability
- **Extension of general-purpose language, e.g. GASP V**
  - —library of routines for common simulation tasks
- **Simulation package, e.g.**
  - —RSIM, NS-2 network simulator
  - —Hyperformix workbench (http://www.hyperformix.com)

Simulation Types

- **Monte Carlo**
- **Trace-driven simulation**
- **Program or Execution-driven simulation**

Monte-Carlo Simulations

- **Model probabilistic phenomenon that do not change over time**
- **Applications**
  - —simulation of random or stochastic processes
    - – complex physical phenomena such as radiation transport
    - – sub-nuclear processes in high energy physics experiments
    - – traffic flow
  - —evaluation of integrals
- **Requirements**
  - —system can be described by probability density functions
  - —good pseudo-random number generator available
- **How they are commonly performed**
  - —given PDFs, simulations proceed by random sampling
  - —multiple simulations (trials) are performed
  - —desired result is taken as avg over # of observations
  - —predictions of variance in avg result used to estimate #trials needed to achieve a given error bound

Trace-driven Simulation

- Trace = time ordered record of events on real system
- Applications: analyze paging, scheduling, caches, etc.
- Advantages
  - —credibility: easy to sell
  - —easy validation: compare with measured system
  - —accurate workload: trace preserves correlation & interference effects
  - —detailed tradeoffs: possible to evaluate small changes in model
  - —less randomness: deterministic input reduces output randomness
  - —fair comparison of alternatives
  - —similarity of implementation: model is similar to system
- Disadvantages
  - —complexity: requires detailed simulation of system
  - —representativeness: traces from one system may not be representative
  - —finiteness: may not represent much time because of size constraints
  - —single point of validation: algorithm good for one trace, not others
  - —high level of detail: simulations can be costly
  - —hard to evaluate changes in workload characteristics: need another trace
    - – no feedback from simulation of changes that effect event ordering

Program and Execution-driven Simulation

- Example: SimpleScalar [Austin, Larsen, Ernst 2002]
- Similar to trace-driven simulation except
  —program under study and simulation are interleaved
    – produce and consume event stream in interleaved fashion
- Key advantages over trace-driven simulation
  —avoids specialized hardware for collecting traces
  —avoids storage of long traces
  —simpler to study new workloads

Discrete-Event Simulations

- Discrete-event simulations use discrete-state model of system
  - —e.g. model number of threads queued for various resources
  - —may use discrete or continuous time values
- Components
  - —event scheduler: linked list of pending events
    - – operations: schedule event X at time T; hold event X for time interval dt; cancel previously scheduled event X; hold X indefinitely; schedule indefinitely held event
  - —simulation clock: maintains global time
    - – unit time or event-driven advancement
  - —system state variables
  - —event routines: one for each event type
  - —input routines: read model parameters
  - —initialization routines for system variables & RNG
  - —trace routines: print intermediate results periodically
  - —dynamic memory management, usually GC managed storage
  - —report generator to calculate and print final result
  - —main program: invokes all components in proper order

Event Sets for Discrete-Event Simulations

- Ordered set of future event notices
  - —typically an ordered linked list
- Operations
  - —insert event
  - —find next scheduled event
  - —remove next scheduled event
- Choice of data structure affects execution time
  - —best depends on frequency of insertion/deletion and avg # events
- Possible implementations
  - —ordered doubly-linked list (used in SIMULA, GASP, GPSS)
  - —indexed linear list
    - – divide future into indexed intervals of Δt
    - – each interval has own sublist
  - —tree structures, e.g. heap
  - —skip list

Thought Questions

- **Which type of simulation should be used for each of the following?**
  - —to model destination address reference patterns in network traffic given that the pattern depends on a large # of factors
  - —to model scheduling in a multiprocessor system given that the request arrivals have a known distribution
  - —to determine the value of π
- **Why is the unit-time approach usually not used?**