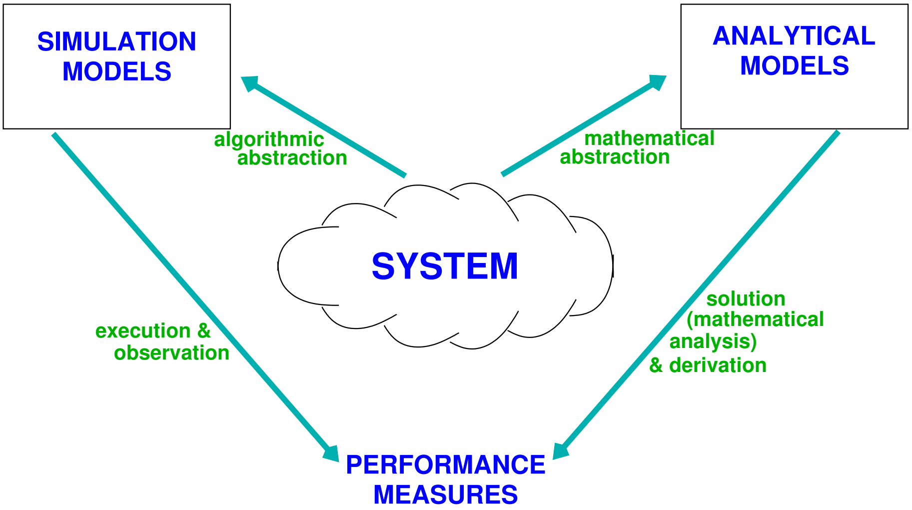
Advanced Topics in Software Engineering: Simulation

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Introduction



Assumptions

- We still assume that the system is characterised by a family of random variables $\{X(t), t \in T\}$.

Assumptions

- We still assume that the system is characterised by a family of random variables $\{X(t), t \in T\}$.
- As the value of time increases, and in response to the "environment" (represented by random variables within the model) the stochastic process progresses from state to state.

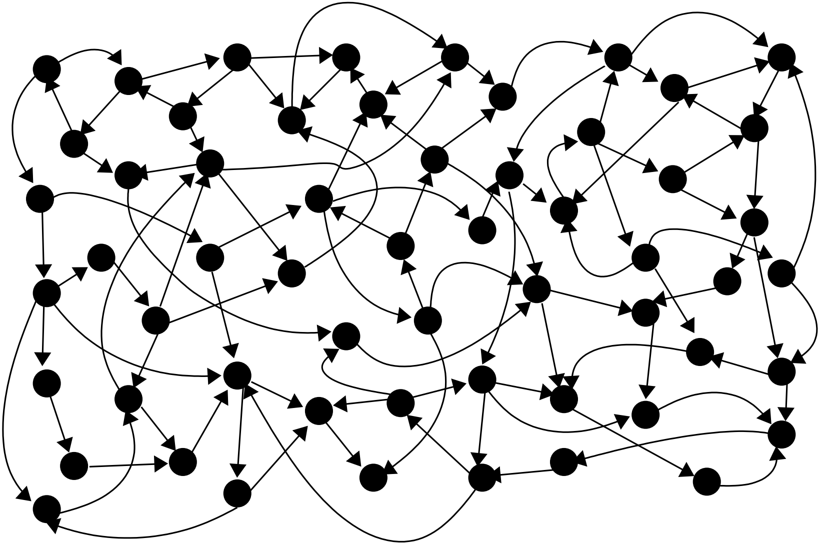
Assumptions

- We still assume that the system is characterised by a family of random variables $\{X(t), t \in T\}$.
- As the value of time increases, and in response to the "environment" (represented by random variables within the model) the stochastic process progresses from state to state.
- Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in a state space, S, its position at time t being $X(t)$.

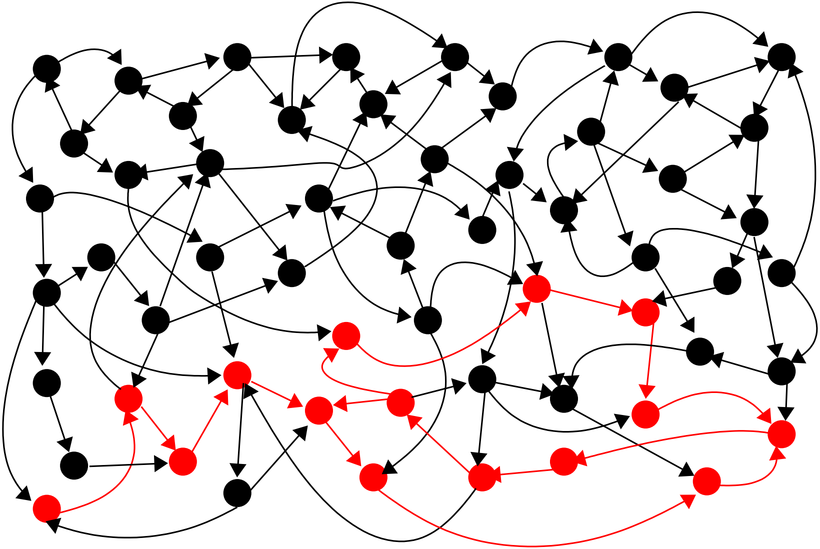
Assumptions

- We still assume that the system is characterised by a family of random variables $\{X(t), t \in T\}$.
- As the value of time increases, and in response to the "environment" (represented by random variables within the model) the stochastic process progresses from state to state.
- Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in a state space, S, its position at time t being $X(t)$.
- These paths are called sample paths.

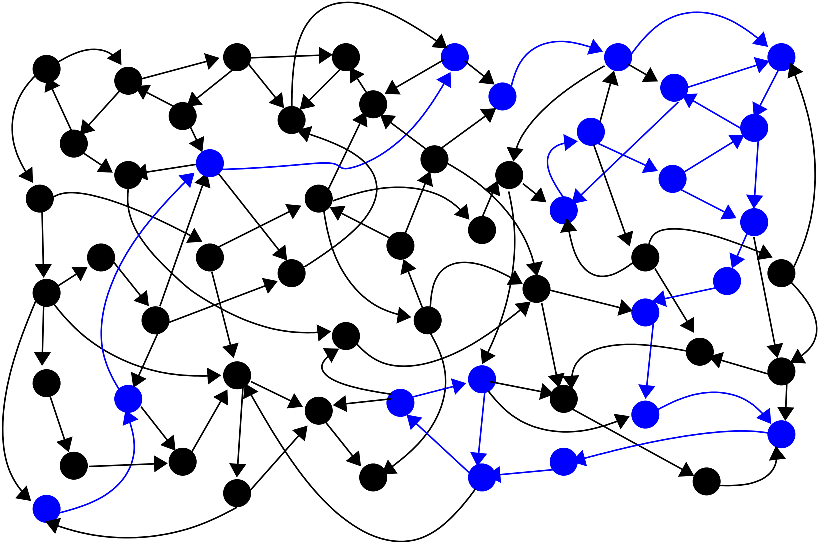
State space and sample paths



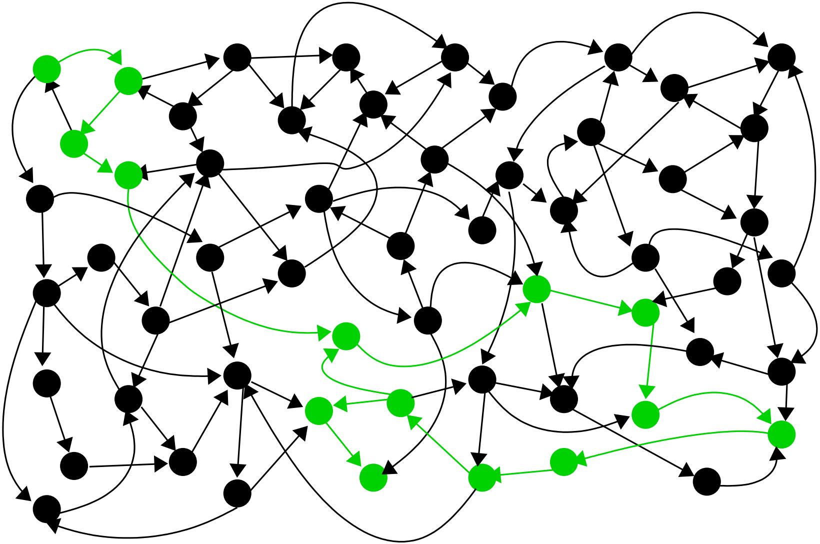
State space and sample paths



State space and sample paths



State space and sample paths



Sample paths and runs

- Using the analytic approach of Markov processes we characterised all possible sample paths by the global balance equations.

Sample paths and runs

- Using the analytic approach of Markov processes we characterised all possible sample paths by the global balance equations.
- Using simulation we investigate the sample paths directly.

Sample paths and runs

- Using the analytic approach of Markov processes we characterised all possible sample paths by the global balance equations.
- Using simulation we investigate the sample paths directly.
- We allow the model to trace out a sample path over the state space.

Sample paths and runs

- Using the analytic approach of Markov processes we characterised all possible sample paths by the global balance equations.
- Using simulation we investigate the sample paths directly.
- We allow the model to trace out a sample path over the state space.
- Each run of the simulation model will generate another, usually distinct, sample path.

Benefits of simulation

There are a variety of reasons why simulation may be preferable to analytical modelling:

Level of Abstraction

- Markovian modelling relies on many assumptions and abstractions which may not be appropriate for the system being studied.
- It may be unrealistic to assume that only one event can happen at any time, or that the inter-event times are all exponentially distributed.
- Simulation models allow us to represent a system at arbitrary levels of detail. This can also be a disadvantage since elaborate models take a long time to run and produce statistically significant output.

Benefits of simulation

There are a variety of reasons why simulation may be preferable to analytical modelling:

Transient Analysis

- In some cases we are not interested in the steady state behaviour of a system, but in its transient behaviour.
- Some systems never reach a steady state. Those that do usually have a "warm-up" period while the behaviour settles into the regular pattern which characterises steady state.
- The analytic solutions ignore this period since the global balance equations only capture the behaviour after steady state has been reached.
- A sample path derived from a simulation model will clearly represent transient behaviour in addition to steady state.

Benefits of simulation

There are a variety of reasons why simulation may be preferable to analytical modelling:

Size of State Space

- Generally solving a model analytically involves constructing and storing the complete state space of the model.
- For a Markov process with N states solving the global balance equations involves (at least) an $N \times N$ matrix ($\mathbf{Q}$) and a vector with N elements (π).
- When N becomes very large this becomes infeasible.
- In contrast, in a simulation model the state space is generated "on-the-fly" by the model itself during execution so it does not need to be all stored at once.

Constructing simulation models

- Simulation models are complex computer programs. They can be programmed directly in any programming language but there are distinct advantages to using a package specifically designed for simulation.

Constructing simulation models

- Simulation models are complex computer programs. They can be programmed directly in any programming language but there are distinct advantages to using a package specifically designed for simulation.
- Simulation packages such as Stochastic Simulation in Java (SSJ) provide facilities for many of the routine features of a simulation model. These features are common to all models, regardless of the system being represented.

Constructing simulation models

- Simulation models are complex computer programs. They can be programmed directly in any programming language but there are distinct advantages to using a package specifically designed for simulation.
- Simulation packages such as Stochastic Simulation in Java (SSJ) provide facilities for many of the routine features of a simulation model. These features are common to all models, regardless of the system being represented.
- This allows the performance analyst to concentrate on the issues specific to the system being modelled and to not worry about issues which are general to all simulations.

Simulation management

Some of the common features of simulation management are listed below.

- Event scheduler
- Simulation clock and time management
- System state variables
- Event routines
- Random number/random variate generator
- Report generator
- Trace routines
- Dynamic memory management

Event scheduler

An event scheduler keeps track of the events which are waiting to happen, usually as a linked list, and allows them to be manipulated in various ways. For example,

- schedule event E at time T;
- hold event E for a time interval ∂t;
- cancel a previously scheduled event E;
- hold event E indefinitely (until it is scheduled by another event);
- schedule an indefinitely held event.

Event scheduler

An event scheduler keeps track of the events which are waiting to happen, usually as a linked list, and allows them to be manipulated in various ways. For example,

- schedule event E at time T;
- hold event E for a time interval ∂t;
- cancel a previously scheduled event E;
- hold event E indefinitely (until it is scheduled by another event);
- schedule an indefinitely held event.

Event scheduler must be efficient

The event scheduler is called before every event, and it may be called several times during one event to schedule other new events.

Simulation clock and time management

- Every simulation model must have a global variable representing the simulated time.
- The event scheduler is usually responsible for advancing this time, either one unit at a time or, more commonly, directly to the time of the next scheduled event.
- This latter approach is called event-driven time management.

System state variables

- Since a simulation model generates a random walk over the state space of the system it is essential that the model has variables to capture the state of the system at each step.
- If a simulation run is stopped in the middle, it can be restarted later if, and only if, the values of all state variables are known.

Event routines

- Each event in the system brings about a state change.
- In the simulation model the effect of each event must be represented in a way which updates the system state variables, and in some cases, schedules other events.
- How the event routines are generated will depend on the simulation modelling paradigm used to construct the model.

Random number/random variate generator

- Random numbers play a crucial role in most discrete event simulations.
- A random number generator is used to generate a sequence of random values between 0 and 1.
- These values are then transformed to produce a sequence of random values which satisfy the desired distribution. This second step is sometimes called random variate generation.

Random number/random variate generator

- Random numbers play a crucial role in most discrete event simulations.
- A random number generator is used to generate a sequence of random values between 0 and 1.
- These values are then transformed to produce a sequence of random values which satisfy the desired distribution. This second step is sometimes called random variate generation.

Example

The impact of the environment on the system, e.g. inter-arrival times, is usually represented by random variables of some specified distribution.

Report generator

- Performance measures are derived from a simulation run by observing the values of parameters of interest during the execution.
- Most simulation modelling languages and packages contain built-in routines to calculate statistics from these observations and generate a report when the run is completed.

Trace routines

- A trace of the system can be a useful tool for debugging (sometimes called verifying) and validating a model.
- It is a time-ordered list of events, state variable values or output parameter values.
- Most simulation languages provide routines to generate traces which can be switched on or off in a particular run of the model.

Trace routines

- A trace of the system can be a useful tool for debugging (sometimes called verifying) and validating a model.
- It is a time-ordered list of events, state variable values or output parameter values.
- Most simulation languages provide routines to generate traces which can be switched on or off in a particular run of the model.

Note

Since trace generation is usually very inefficient it is generally only used during model development.

Dynamic memory management

- The number of active entities during the execution of a simulation model will vary continuously as new entities are created and old ones become obsolete.
- Most simulation languages provide automatic garbage collection to remove obsolete entities.

Approaches to simulation

- There are a number of approaches to discrete event simulation; the two most commonly used are event based modelling and process based modelling.

Approaches to simulation

- There are a number of approaches to discrete event simulation; the two most commonly used are event based modelling and process based modelling.
- Note that the high level modelling paradigms which we have already considered in the course—Population models and PEPA—can be used to generate simulation models as well as Markov processes.

Event-based Simulation

- Event-based simulation focusses the modeller's attention on the individual events which can occur within the system.

Event-based Simulation

- Event-based simulation focusses the modeller's attention on the individual events which can occur within the system.
- An event within the system may generate several actions in the model—these are grouped together in an event routine.

Event-based Simulation

- Event-based simulation focusses the modeller's attention on the individual events which can occur within the system.
- An event within the system may generate several actions in the model—these are grouped together in an event routine.
- The event scheduler maintains a pointer to the appropriate event routine, and this is executed when the event reaches the head of the event list.

Example: A simple queue

- A good example of a system suited to event-based simulation would be a simple queue with deterministic arrival times, deterministic service times and customer defections at fixed times.

Example: A simple queue

- A good example of a system suited to event-based simulation would be a simple queue with deterministic arrival times, deterministic service times and customer defections at fixed times.
- The events are a customer arrives, A; a customer defects, D; a customer begins service, B; and a customer ends service, E.

Example: A simple queue

- A good example of a system suited to event-based simulation would be a simple queue with deterministic arrival times, deterministic service times and customer defections at fixed times.
- The events are a customer arrives, A; a customer defects, D; a customer begins service, B; and a customer ends service, E.
- At each event time as well as the processing of the event to represent the behaviour of the system, some processing internal to the model might be done.

Example: A simple queue

- A good example of a system suited to event-based simulation would be a simple queue with deterministic arrival times, deterministic service times and customer defections at fixed times.
- The events are a customer arrives, A; a customer defects, D; a customer begins service, B; and a customer ends service, E.
- At each event time as well as the processing of the event to represent the behaviour of the system, some processing internal to the model might be done.
- This can be event tracing or statistical collection. For example, calculating the average queue length and throughput.

Events: a customer arrives, A

An event A at time t

- add one to the state variable representing queue length, and record the time at which the change occurred,
- schedule an event D at the time $t + d$, where d is the length of time a customer will wait without defecting,
- schedule an event B to occur as soon as possible, depending on the availability of the server,
- schedule another event A at time $t + a$ where a is the inter-arrival time.

Events: a customer defects, D

An event D at time $t + d$

- decrease the queue length by one and the record the time at which the change occurred,
- de-schedule event B on hold since time t.

Events: a customer begins service, B

An event B at time $t + w$ ($w < d$)

- decrease the queue length by one and record the time at which the change occurred,
- de-schedule the event D at time $t + d$,
- schedule an event E at time $t + w + s$, where s is the service time.

Events: a customer ends service, E

An event E at time $t + w + s$

- increment the busy time of the service centre by s,
- add one to the total number of customers served,
- activate the first event B waiting in the event list.

Illustration

T = 0

A(0)

↓

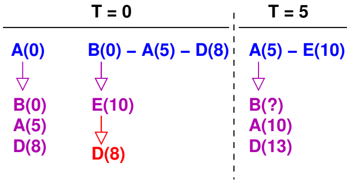
B(0)
A(5)
D(8)

Illustration

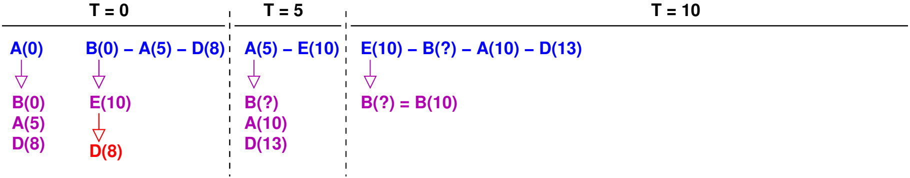
T = 0

A(0)	B(0) – A(5) – D(8)
↓	↓
B(0) A(5) D(8)	E(10)
	↓
	D(8)

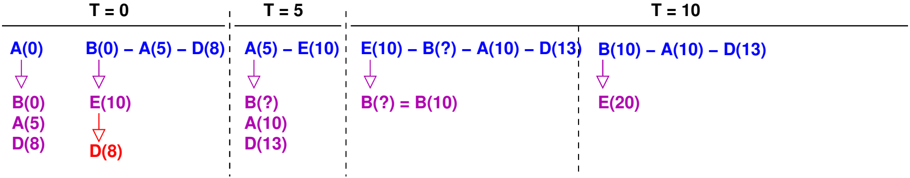
Illustration



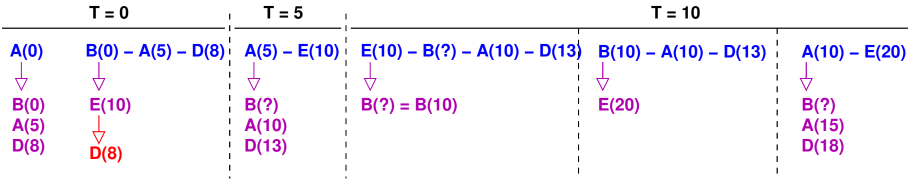
Illustration



Illustration



Illustration



Illustration

T = 0

A(0)

B(0)
A(5)
D(8)

B(0) – A(5) – D(8)

E(10)

D(8)

T = 5

A(5) – E(10)

B(?)
A(10)
D(13)

T = 10

E(10) – B(?) – A(10) – D(13)

B(?) = B(10)

B(10) – A(10) – D(13)

E(20)

A(10) – E(20)

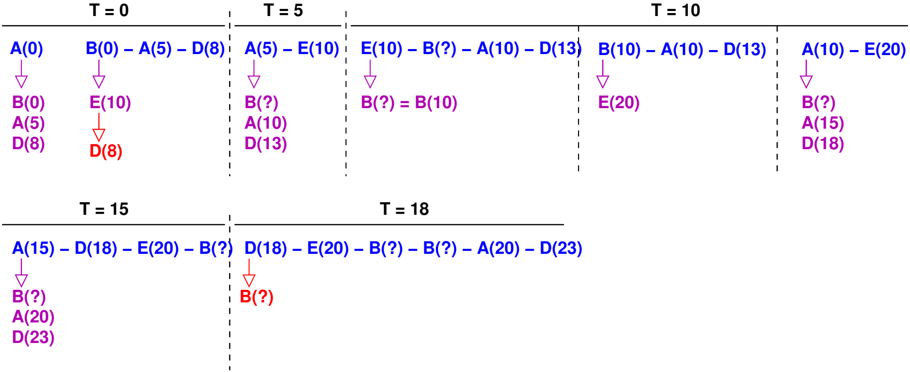
B(?)
A(15)
D(18)

T = 15

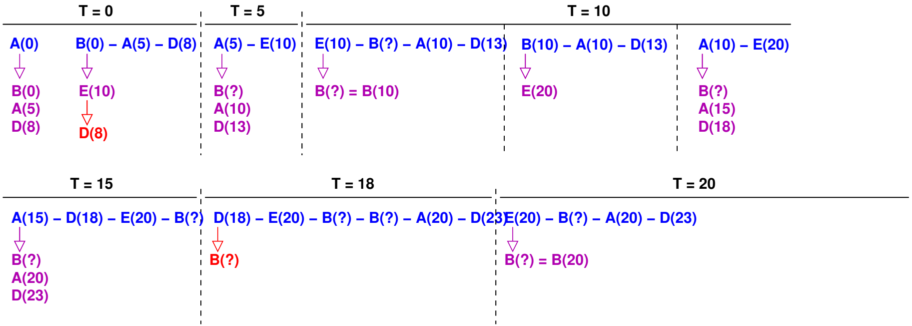
A(15) – D(18) – E(20) – B(?)

B(?)
A(20)
D(23)

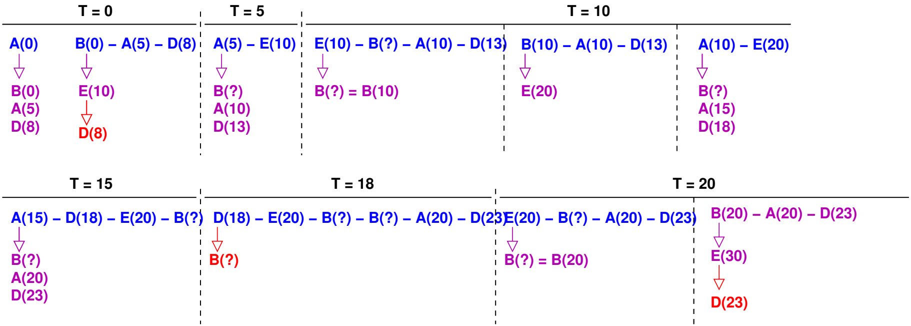
Illustration



Illustration



Illustration



Process-based Simulation

- The process-based approach to simulation modelling collects events together into related sequences which are ordered by time.

Process-based Simulation

- The process-based approach to simulation modelling collects events together into related sequences which are ordered by time.
- These sequences are related in the sense that they all involve the same entity within the system; they are termed processes.

Process-based Simulation

- The process-based approach to simulation modelling collects events together into related sequences which are ordered by time.
- These sequences are related in the sense that they all involve the same entity within the system; they are termed processes.
- For example, above we could consider each customer to be a process within the system, since it generates a sequence of related events, and track its progress through the queue.

Processes and life-cycles

- The process-based approach views a system as a web of concurrent, interacting entities—the processes.

Processes and life-cycles

- The process-based approach views a system as a web of concurrent, interacting entities—the processes.
- All the actions associated with an entity's behaviour are grouped together to form a life-cycle for entities of that type.

Processes and life-cycles

- The process-based approach views a system as a web of concurrent, interacting entities—the processes.
- All the actions associated with an entity's behaviour are grouped together to form a life-cycle for entities of that type.
- The event scheduler still maintains a list of scheduled events centrally but this will now generally be in the form of a pointer to a process/object.

Processes and life-cycles

- The process-based approach views a system as a web of concurrent, interacting entities—the processes.
- All the actions associated with an entity's behaviour are grouped together to form a life-cycle for entities of that type.
- The event scheduler still maintains a list of scheduled events centrally but this will now generally be in the form of a pointer to a process/object.
- The process will maintain a record of its current state and which action it should perform when next scheduled.

Processes and life-cycles

- The process-based approach views a system as a web of concurrent, interacting entities—the processes.
- All the actions associated with an entity's behaviour are grouped together to form a life-cycle for entities of that type.
- The event scheduler still maintains a list of scheduled events centrally but this will now generally be in the form of a pointer to a process/object.
- The process will maintain a record of its current state and which action it should perform when next scheduled.
- This style of modelling maps well to object-oriented programming: a class is associated with each type of entity; objects then represent instances of the entity.

Example: A simple queue

- For the example of a queue with deterministic inter-arrival and service times (but no defections), we could define a class to represent the arrival process.

Example: A simple queue

- For the example of a queue with deterministic inter-arrival and service times (but no defections), we could define a class to represent the arrival process.
- This class (`Source`) would generate the event representing a customer entering the queue and then delay until the arrival time of the next customer.

Example: A simple queue

- For the example of a queue with deterministic inter-arrival and service times (but no defections), we could define a class to represent the arrival process.
- This class (`Source`) would generate the event representing a customer entering the queue and then delay until the arrival time of the next customer.
- A second class would represent the server (`Server`).

Example: A simple queue

- For the example of a queue with deterministic inter-arrival and service times (but no defections), we could define a class to represent the arrival process.
- This class (`Source`) would generate the event representing a customer entering the queue and then delay until the arrival time of the next customer.
- A second class would represent the server (`Server`).
- This is passive in the sense that it first waits to be notified of an event (the arrival of a customer) and then represents the service of the customer as a delay.

Common mistakes in simulation studies

- Inappropriate level of detail
  Because arbitrary levels of detail are possible it is sometimes tempting to represent too much in the model. This will have a cost in terms of execution time.

Common mistakes in simulation studies

- Inappropriate level of detail
  Because arbitrary levels of detail are possible it is sometimes tempting to represent too much in the model. This will have a cost in terms of execution time.
- Unverified models
  Simulation models are complex programs and as such are prone to bugs in the same way that any complex program is. Verification is intended to make sure that the model behaves as it was intended to.

Common mistakes in simulation studies

- Inappropriate level of detail
  Because arbitrary levels of detail are possible it is sometimes tempting to represent too much in the model. This will have a cost in terms of execution time.
- Unverified models
  Simulation models are complex programs and as such are prone to bugs in the same way that any complex program is. Verification is intended to make sure that the model behaves as it was intended to.
- Invalid models
  Validation is needed ensure that the model is a good representation of the system. A model may be bug-free but still be incorrect in the sense that it is based on invalid assumptions.

Common mistakes in simulation studies

- Too short simulation runs
  A large model must be executed for a long (simulated) time to ensure that the sample path which is generated is statistically valid.

Common mistakes in simulation studies

- Too short simulation runs
  A large model must be executed for a long (simulated) time to ensure that the sample path which is generated is statistically valid.
- Single simulation runs
  Each run of the simulation represents only one sample path based on a particular sequence of random numbers. In order for results to be statistically valid they should be based on several sample paths obtained using different sequences.

Common mistakes in simulation studies

- Too short simulation runs
  A large model must be executed for a long (simulated) time to ensure that the sample path which is generated is statistically valid.
- Single simulation runs
  Each run of the simulation represents only one sample path based on a particular sequence of random numbers. In order for results to be statistically valid they should be based on several sample paths obtained using different sequences.
- Poor random-number generators
  Random number generators are used extensively in simulation models. A poor random-number generator may introduce correlation and/or bias into the value of those random variables.

To be continued...

Advanced Topics in Software Engineering: Random Variables and Simulation

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Introduction

- Random variables play two important roles in simulation models.

Introduction

- Random variables play two important roles in simulation models.
    1. We assume that within our models some delays will not have deterministic values, but instead will be represented by random variables; and

Introduction

- Random variables play two important roles in simulation models.
  1. We assume that within our models some delays will not have deterministic values, but instead will be represented by random variables; and
  2. when a choice must be made within the behaviour of an entity we will sometimes want the decision to be made probabilistically.

Introduction

- Random variables play two important roles in simulation models.
  1. We assume that within our models some delays will not have deterministic values, but instead will be represented by random variables; and
  2. when a choice must be made within the behaviour of an entity we will sometimes want the decision to be made probabilistically.
- Both cases will involve sampling a probability distribution to extract a value each time this part of the entity's behaviour is reached.

Introduction

- Random variables play two important roles in simulation models.
  1. We assume that within our models some delays will not have deterministic values, but instead will be represented by random variables; and
  2. when a choice must be made within the behaviour of an entity we will sometimes want the decision to be made probabilistically.
- Both cases will involve sampling a probability distribution to extract a value each time this part of the entity's behaviour is reached.
- Both cases rely on the random number generator.

Random variables

- We also assume that the variables characterising the behaviour of the system/model, the performance measures or output parameters, are also random variables.

Random variables

- We also assume that the variables characterising the behaviour of the system/model, the performance measures or output parameters, are also random variables.
- In general, each run of the simulation model provides a single estimate for these random variables.

Simulation and steady-state

- If we want steady state values, the longer we run a simulation the better our estimate will be. However, it still remains a single observation in the sample space.

Simulation and steady-state

- If we want steady state values, the longer we run a simulation the better our estimate will be. However, it still remains a single observation in the sample space.
- We need more than a single estimate in order to draw conclusions about the system.

Simulation and steady-state

- If we want steady state values, the longer we run a simulation the better our estimate will be. However, it still remains a single observation in the sample space.
- We need more than a single estimate in order to draw conclusions about the system.
- We use output analysis techniques to improve the quality of an estimates and to develop ways of gaining more observations without excessive computational cost.

Simulation and steady-state

- If we want steady state values, the longer we run a simulation the better our estimate will be. However, it still remains a single observation in the sample space.
- We need more than a single estimate in order to draw conclusions about the system.
- We use output analysis techniques to improve the quality of an estimates and to develop ways of gaining more observations without excessive computational cost.
- Realistic simulation models take a long time to run—there is always a trade-off between accuracy of estimates and execution time.

Random number generation

- Generating random values for variables with a specified random distribution, such as an exponential or normal distribution, involves two steps.

Random number generation

- Generating random values for variables with a specified random distribution, such as an exponential or normal distribution, involves two steps.
    1. A sequence of random numbers distributed uniformly between 0 and 1 is obtained.

Random number generation

- Generating random values for variables with a specified random distribution, such as an exponential or normal distribution, involves two steps.
    1. A sequence of random numbers distributed uniformly between 0 and 1 is obtained.
    2. The sequence is transformed to produce a sequence of random values which satisfy the desired distribution.

Random number generation

- Generating random values for variables with a specified random distribution, such as an exponential or normal distribution, involves two steps.
  1. A sequence of random numbers distributed uniformly between 0 and 1 is obtained.
  2. The sequence is transformed to produce a sequence of random values which satisfy the desired distribution.
- This second step is called random variate generation.

Generating uniform random numbers

- To obtain a sequence of uniform random numbers between 0 and 1 in fact we generate a sequence X_k of integers in the range $[0, M - 1]$

Generating uniform random numbers

- To obtain a sequence of uniform random numbers between 0 and 1 in fact we generate a sequence X_k of integers in the range $[0, M-1]$
- The sequence $X_k/(M-1)$ will then be approximately uniformly distributed over $(0, 1)$.

Generating uniform random numbers

- To obtain a sequence of uniform random numbers between 0 and 1 in fact we generate a sequence X_k of integers in the range $[0, M-1]$
- The sequence $X_k/(M-1)$ will then be approximately uniformly distributed over $(0,1)$.
- In 1951, D.H. Lehmer discovered that the residues of successive powers of a number have good randomness properties.

Lehmer generators

- Lehmer obtained the kth number in the sequence by dividing the kth power of an integer a by another integer M and taking the remainder.

$$X_k = a^k \bmod M$$

Lehmer generators

- Lehmer obtained the kth number in the sequence by dividing the kth power of an integer a by another integer M and taking the remainder.

$$X_k = a^k \bmod M$$

This can be expressed as an iteration:

$$X_k = (a \times X_{k-1}) \bmod M$$

Lehmer generators

- Lehmer obtained the kth number in the sequence by dividing the kth power of an integer a by another integer M and taking the remainder.

  $$X_k = a^k \bmod M$$

  This can be expressed as an iteration:

  $$X_k = (a \times X_{k-1}) \bmod M$$

- The parameters a and M are called the multiplier and the modulus respectively.

Lehmer generators

- Lehmer obtained the kth number in the sequence by dividing the kth power of an integer a by another integer M and taking the remainder.

  $$X_k = a^k \bmod M$$

  This can be expressed as an iteration:

  $$X_k = (a \times X_{k-1}) \bmod M$$

- The parameters a and M are called the multiplier and the modulus respectively.
- Random number generators of this form are called Lehmer generators, or multiplicative linear-congruential generators.

Desirable properties for a random number generator

- It should be efficiently computable
  Simulations typically require several thousand random numbers in each run so processor time to generate these should be kept small.

Desirable properties for a random number generator

- It should be efficiently computable
  Simulations typically require several thousand random numbers in each run so processor time to generate these should be kept small.
- It should be pseudo-random
  Given the same seed, the random number generator should produce exactly the same sequence of numbers. (Good for reproducibility of experiments.)

Desirable properties for a random number generator

- It should be efficiently computable
  Simulations typically require several thousand random numbers in each run so processor time to generate these should be kept small.
- It should be pseudo-random
  Given the same seed, the random number generator should produce exactly the same sequence of numbers. (Good for reproducibility of experiments.)
- The cycle should be long
  A short cycles may lead to repeated event sequences. This may limit the useful length of simulation runs.

Desirable properties for a random number generator

- It should be efficiently computable
  Simulations typically require several thousand random numbers in each run so processor time to generate these should be kept small.
- It should be pseudo-random
  Given the same seed, the random number generator should produce exactly the same sequence of numbers. (Good for reproducibility of experiments.)
- The cycle should be long
  A short cycles may lead to repeated event sequences. This may limit the useful length of simulation runs.
- Independent and uniformly distributed succesive values
  The correlation between successive numbers should be small.

Problems with random number generators

- Research has shown that Lehmer generators obey these properties provided a and M are carefully chosen. However care is needed.
- In the early 1970s most university mainframes were using a linear-congruence generator known as `RANDU`.
- It used the values $a = 65539$ and $M = 2^{31}$.
- Although the output looked random, detailed statistical analysis showed that there was significant correlation in the output.

Efficiency of random number generators

- This form of generator continues to be used, if somewhat more warily.
- These generators are particularly efficient if M is chosen to be a power of 2.
- In this case finding the residue amounts to simply truncating the result of the multiplication.
- However a modulus of the form 2^k results in a shorter cycle: 2^{k-2} at best.

Mersenne Twister

- One of the best families of random number generators for simulation is that based on the Mersenne Twister algorithm.
- It is used by default in python, R, MATLAB and several other languages.
- It comes in a number of variants, but the commonly used MT19937 variant produces a sequence of 32-bit integers, and has the following desirable properties:
  - It has a very long period of $2^{19937} - 1$.
  - It passes numerous tests for statistical randomness, including some stringent tests which are failed by linear congruential random number generators.

Random variate generation algorithms

Random variate generation algorithms for values of the commonly used probability distributions, based on a uniformly distributed stream of values between 0 and 1, can be found in many books on simulation and performance modelling.

Random variate generation algorithms

Random variate generation algorithms for values of the commonly used probability distributions, based on a uniformly distributed stream of values between 0 and 1, can be found in many books on simulation and performance modelling.

A good example is the book by Raj Jain:

The Art of Computer Systems Performance Analysis, Wiley, 1991.

Inverse transformations

- Inverse transformation algorithms are based on the observation that for any probability distribution with distribution function $F(x)$, the value of $F(x)$ is uniformly distributed between 0 and 1.

Inverse transformations

- Inverse transformation algorithms are based on the observation that for any probability distribution with distribution function $F(x)$, the value of $F(x)$ is uniformly distributed between 0 and 1.
- Thus, using values from the random number stream, $u = X_k$, the function is inverted to find the next value of x: $x = F^{-1}(u)$.

Exponential distributions

For example, given a random number u, we generate the next value in an exponential distribution with parameter λ as

$$x = -\frac{1}{\lambda} \ln(u)$$

Exponential distributions

For example, given a random number u, we generate the next value in an exponential distribution with parameter λ as

$$x = -\frac{1}{\lambda}\ln(u)$$

Note

Strictly speaking, the equation should be

$$x = -\frac{1}{\lambda}\ln(1-u)$$

but since u is uniformly distributed between 0 and 1, $1-u$ will be uniformly distributed between 0 and 1 and the generation algorithm can be simplified.

Boolean-valued distributions

- Boolean-valued distributions which are used to make decisions within a model take a single real parameter, p, such that $0 \leq p \leq 1$.
- This represents the probability of a "positive" outcome.
- Then each time the branching point in the model is reached, the next random number in the stream is generated $u = X_k$.
- If $u \leq p$ the positive branch is taken;
- If $u > p$ the other branch is selected.

Simulation packages

- One of the benefits of using a simulation package is that at least some of these algorithms are provided for us.
- Each time that a distribution is instantiated the seed for the random number generator can be set explicitly.
- If seeds are not well-spaced there may be overlap between the sequences of random numbers used by the generators resulting in correlation between the samples used in the simulation.
- Some simulation packages provide an automatic seeding mechanism which will seed each distribution with a distinct seed which is far in the cycle from other seeds currently in use.

Simulation output analysis

- Our objective in constructing a simulation model is to generate one or more performance measures for the system.

Simulation output analysis

- Our objective in constructing a simulation model is to generate one or more performance measures for the system.
- In the Markov models such measures were derived from the steady state probability distribution, after the model solution.

Simulation output analysis

- Our objective in constructing a simulation model is to generate one or more performance measures for the system.
- In the Markov models such measures were derived from the steady state probability distribution, after the model solution.
- In contrast, in a simulation model measures are observed or evaluated directly during the execution of the model.

Simulation output analysis

- Our objective in constructing a simulation model is to generate one or more performance measures for the system.
- In the Markov models such measures were derived from the steady state probability distribution, after the model solution.
- In contrast, in a simulation model measures are observed or evaluated directly during the execution of the model.
- It is part of model construction to make sure that all the necessary counters and updates are in place to allow the measures to be collected as the model runs.

Simulation output analysis

- Our objective in constructing a simulation model is to generate one or more performance measures for the system.
- In the Markov models such measures were derived from the steady state probability distribution, after the model solution.
- In contrast, in a simulation model measures are observed or evaluated directly during the execution of the model.
- It is part of model construction to make sure that all the necessary counters and updates are in place to allow the measures to be collected as the model runs.
- This is sometimes called instrumentation of a model as it is analogous to inserting probes and monitors on a real system.

Simulation trajectories

It is important to remember that each run of a model constitutes a single trajectory over the state space.

Simulation trajectories

It is important to remember that each run of a model constitutes a single trajectory over the state space.

So, in general, any estimate for the value of a performance measure generated from a single run constitutes a single observation in the possible sample space.

Simulation and long-term averages

- To gain an accurate measure of the performance of the system we should not base our results on a single observation.

Simulation and long-term averages

- To gain an accurate measure of the performance of the system we should not base our results on a single observation.
- For steady state analysis the averages we calculate from data collected during execution will always be an approximation of the unknown true long-term averages that characterise the system performance.

Simulation and long-term averages

- To gain an accurate measure of the performance of the system we should not base our results on a single observation.
- For steady state analysis the averages we calculate from data collected during execution will always be an approximation of the unknown true long-term averages that characterise the system performance.
- Important issues are:

Simulation and long-term averages

- To gain an accurate measure of the performance of the system we should not base our results on a single observation.
- For steady state analysis the averages we calculate from data collected during execution will always be an approximation of the unknown true long-term averages that characterise the system performance.
- Important issues are:
  - choosing the starting state of the simulation;

Simulation and long-term averages

- To gain an accurate measure of the performance of the system we should not base our results on a single observation.
- For steady state analysis the averages we calculate from data collected during execution will always be an approximation of the unknown true long-term averages that characterise the system performance.
- Important issues are:
  - choosing the starting state of the simulation;
  - choosing the warm-up period that is allowed to elapse before data collection begins;

Simulation and long-term averages

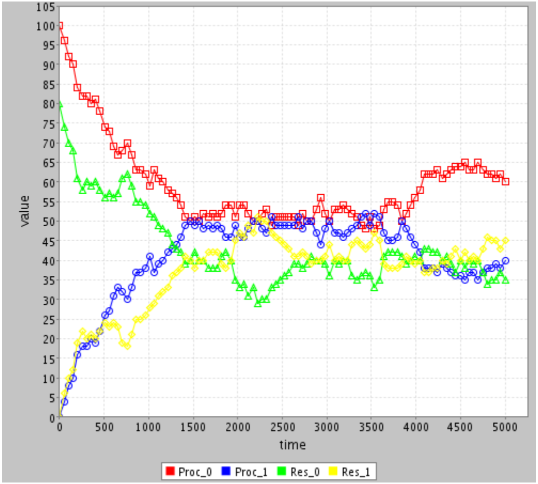
- To gain an accurate measure of the performance of the system we should not base our results on a single observation.
- For steady state analysis the averages we calculate from data collected during execution will always be an approximation of the unknown true long-term averages that characterise the system performance.
- Important issues are:
  - choosing the starting state of the simulation;
  - choosing the warm-up period that is allowed to elapse before data collection begins;
  - choosing a run length that ensures that the calculated averages are representative of the unknown true long term average.

Example

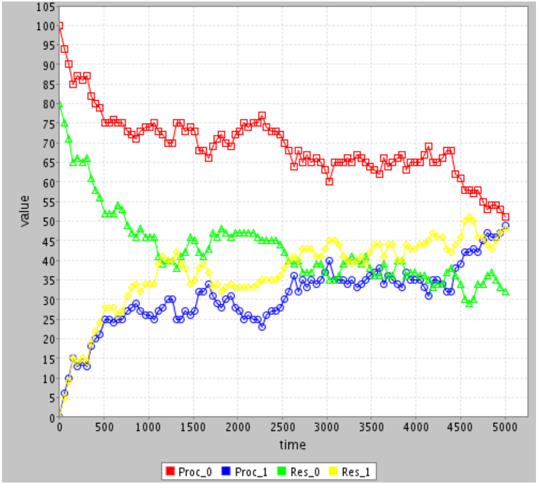
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$
Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]
$$

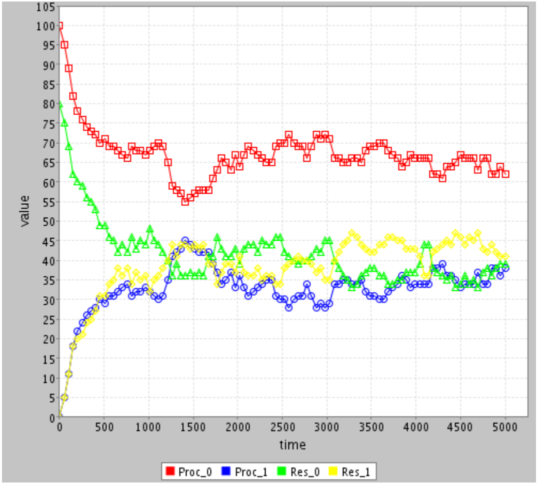
100 processors and 80 resources (simulation run A)



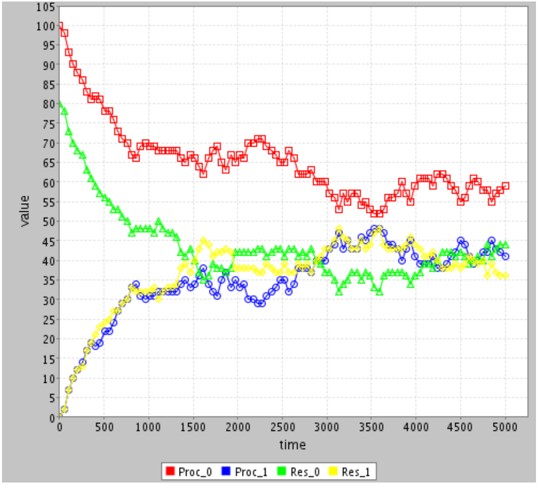
100 processors and 80 resources (simulation run B)



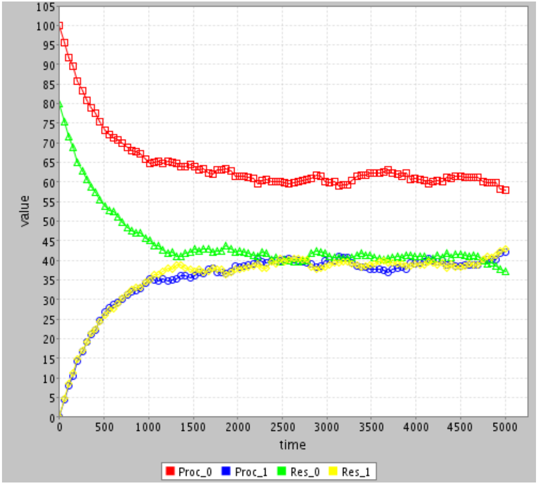
100 processors and 80 resources (simulation run C)



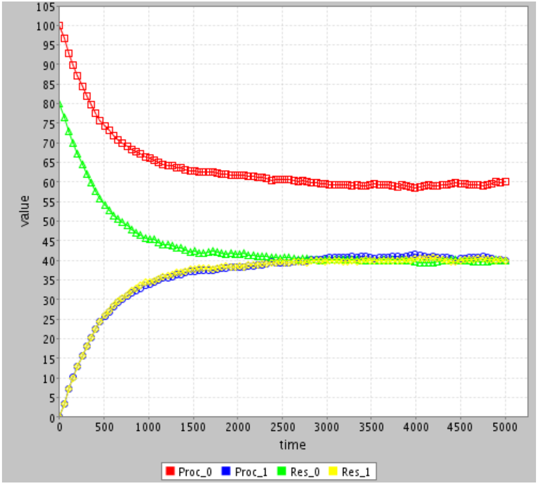
100 processors and 80 resources (simulation run D)



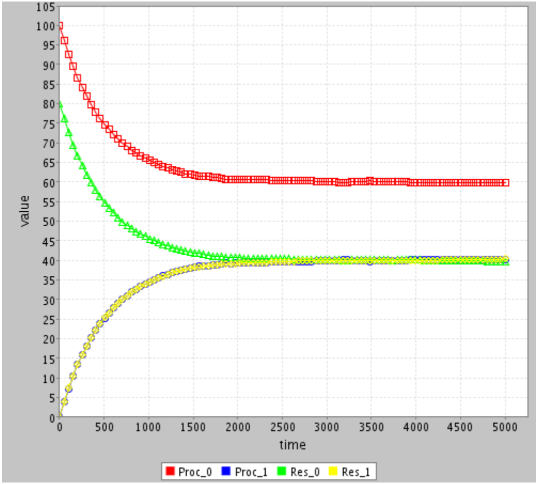
100 processors and 80 resources (average of 10 runs)



100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 runs)



Statistical techniques

- Statistical techniques can be used to assess how and when the calculated averages approximate the true average, i.e. to analyse the accuracy of our current estimate.
- This is often done in terms of a confidence interval.
- A confidence interval expresses probabilistic bounds on the error of our current estimate.

Confidence intervals

A confidence interval (c_1, c_2) with confidence level $X\%$, means that with probability $X/100$ the real value v lies between the values c_1 and c_2, i.e.

$$\Pr(c_1 \leq v \leq c_2) = X/100$$

Confidence intervals

A confidence interval (c_1, c_2) with confidence level $X\%$, means that with probability $X/100$ the real value v lies between the values c_1 and c_2, i.e.

$$Pr(c_1 \leq v \leq c_2) = X/100$$

$X/100$ is usually written in the form $1 - \alpha$, and α is called the significance level, and $(1 - \alpha)$ is called the confidence coefficient.

Confidence intervals and variance

Usually performance modellers will run their simulation models until their observations give them confidence levels of 90% or 95% and a confidence interval which is acceptably tight.

Confidence intervals and variance

Usually performance modellers will run their simulation models until their observations give them confidence levels of 90% or 95% and a confidence interval which is acceptably tight.

Calculation of the confidence interval is based on the variance within the observations which have been gathered.

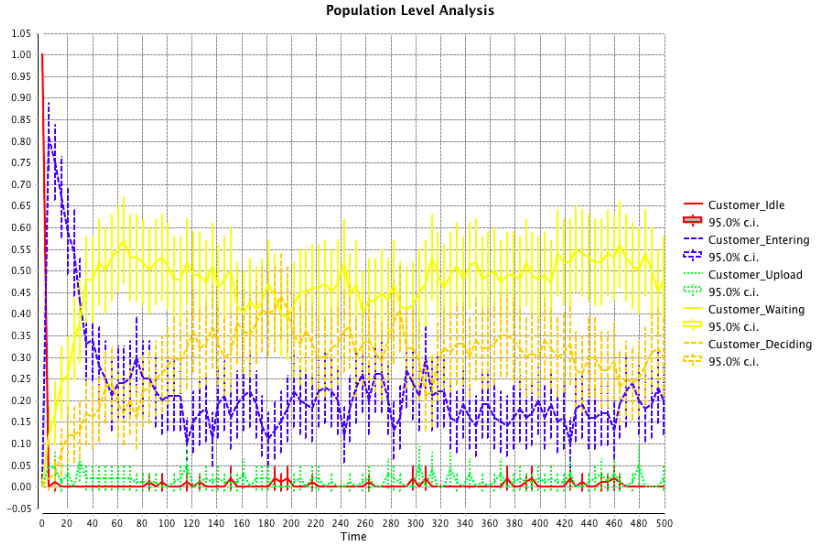
Confidence intervals and variance

Usually performance modellers will run their simulation models until their observations give them confidence levels of 90% or 95% and a confidence interval which is acceptably tight.

Calculation of the confidence interval is based on the variance within the observations which have been gathered.

The greater the variance, the wider the confidence interval; the smaller the variance, the tighter the bounds.

Confidence interval example



Length of simulation runs

For some modelling studies the length of time for which a simulation model should be run is defined by the problem itself.

Length of simulation runs

For some modelling studies the length of time for which a simulation model should be run is defined by the problem itself.

For example, if we wish to investigate how many messages can be processed by a dealers' transaction processing system in the first hour of trading then it makes sense to run the model for 3600 seconds.

Length of simulation runs

For some modelling studies the length of time for which a simulation model should be run is defined by the problem itself.

For example, if we wish to investigate how many messages can be processed by a dealers' transaction processing system in the first hour of trading then it makes sense to run the model for 3600 seconds.

However, if the question is how many messages can be processed in an average hour then running the model for 3600 seconds is unlikely to be enough.

Terminating simulations and cold-start

The first question (“the first hour”) identifies the simulation as a transient or terminating simulation.

Terminating simulations and cold-start

The first question ("the first hour") identifies the simulation as a transient or terminating simulation.

It is said to have a cold-start: the system is initially empty which is not its usual state but we still include this data in the observation period.

Terminating simulations and cold-start

The first question ("the first hour") identifies the simulation as a transient or terminating simulation.

It is said to have a cold-start: the system is initially empty which is not its usual state but we still include this data in the observation period.

For this type of simulation the question becomes how many times the simulation must be repeated (with different random number streams) to achieve a required confidence interval.

Steady-state behaviour

In the second scenario ("an average hour") we are interested in the steady state behaviour of the system.

Steady-state behaviour

In the second scenario ("an average hour") we are interested in the steady state behaviour of the system.

As in Markovian modelling we associate steady state behaviour with long term behaviour.

Steady-state behaviour

In the second scenario (“an average hour”) we are interested in the steady state behaviour of the system.

As in Markovian modelling we associate steady state behaviour with long term behaviour.

In other words we are theoretically interested in the observations obtained from runs of the model which are infinitely long.

Steady-state behaviour

In the second scenario ("an average hour") we are interested in the steady state behaviour of the system.

As in Markovian modelling we associate steady state behaviour with long term behaviour.

In other words we are theoretically interested in the observations obtained from runs of the model which are infinitely long.

However, in practice we are interested in finite run lengths and estimating the steady state distribution of the measures we are interested in from finitely many samples.

Initial conditions, bias

The initial conditions or starting state of a model influence the sequence of states seen in the simulation, especially early in a run.

Initial conditions, bias

The initial conditions or starting state of a model influence the sequence of states seen in the simulation, especially early in a run.

In a steady state distribution the output values should be independent of the starting state.

Initial conditions, bias

The initial conditions or starting state of a model influence the sequence of states seen in the simulation, especially early in a run.

In a steady state distribution the output values should be independent of the starting state.

Thus the modeller must make some effort to remove the effect of the starting state, sometimes termed bias, from the sample data used for estimating the performance measure of interest.

Initial conditions, bias

The initial conditions or starting state of a model influence the sequence of states seen in the simulation, especially early in a run.

In a steady state distribution the output values should be independent of the starting state.

Thus the modeller must make some effort to remove the effect of the starting state, sometimes termed bias, from the sample data used for estimating the performance measure of interest.

Unfortunately there is no precise procedure for this as we cannot generally detect when the model has moved from transient behaviour (the warm-up period) to steady state behaviour.

Heuristics for reducing bias

The common techniques are

1. Long runs.
2. Proper initialisation.
3. Truncation.
4. Initial data deletion.
5. Moving average of independent replications.
6. Batch means.

Heuristics for reducing bias

The common techniques are

1. Long runs.
2. Proper initialisation.
3. Truncation.
4. Initial data deletion.
5. Moving average of independent replications.
6. Batch means.

The last four techniques are all based on the assumption that variability is less during steady state behaviour than during transient behaviour.

Options for terminating a simulation

Option 1

- begin the simulation at time 0
- begin data collection at specified time $w \geq 0$
- complete data collection at specified time $w + t$
- terminate execution of the simulation at time $w + t$
- calculate summary statistics based on sample path data collected in the time interval $(w, w + t)$.

Options for terminating a simulation

Option 2

- begin the simulation at time 0
- begin data collection when the Mth event completes
- complete data collection when the $(M+N)$th event completes
- terminate execution of the simulation when the $(M+N)$th event completes
- calculate summary statistics based on sample path data collected in the time interval (t_M, t_{M+N}), where t_j is the time at which the jth event completes.

Advantages and disadvantages

- Option 1 implies that the simulated time $(w, w+t)$ for data collection is predetermined but the number of event completions is random.
- Conversely, Option 2 implies that the time period for data collection is random but the number of event completions is predetermined.
- In queueing networks, Option 1 is preferable for calculating queue lengths and resource utilisations, whereas Option 2 is preferable for calculating waiting times.

Variance reduction techniques

- Assume that we are running a simulation model in order to estimate some performance measure M.
- During the ith execution of the model we make observations of M, o_{ij} and at the end of the run we calculate the mean value of the observations O_i.
- Note that the observations o_{ij} in most simulations are not independent. Successive observations are often correlated.

Example of correlation

- If we are interested in the delay of messages in a packet-switching network, if the delay of one message is long because the network is heavily congested, the next message is likely to be similarly delayed.

Example of correlation

- If we are interested in the delay of messages in a packet-switching network, if the delay of one message is long because the network is heavily congested, the next message is likely to be similarly delayed.
- Thus the two observations are not independent.

Example of correlation

- If we are interested in the delay of messages in a packet-switching network, if the delay of one message is long because the network is heavily congested, the next message is likely to be similarly delayed.
- Thus the two observations are not independent.

Note

This is why, in general, a simulation model must be run several times.

Independent replications

- If independent replications are used the model is run m times in order to generate m independent observations.

Independent replications

- If independent replications are used the model is run m times in order to generate m independent observations.
- For the runs to be independent, the random number generator seeds must be carefully chosen.

Independent replications

- If independent replications are used the model is run m times in order to generate m independent observations.
- For the runs to be independent, the random number generator seeds must be carefully chosen.
- If steady state or long term behaviour is being investigated the data relating to the warm-up period must be discarded.

Independent replications

- If independent replications are used the model is run m times in order to generate m independent observations.
- For the runs to be independent, the random number generator seeds must be carefully chosen.
- If steady state or long term behaviour is being investigated the data relating to the warm-up period must be discarded.
- Let O denote the mean value of the retained observations, O_i, after m runs.

Independent replications

- If independent replications are used the model is run m times in order to generate m independent observations.
- For the runs to be independent, the random number generator seeds must be carefully chosen.
- If steady state or long term behaviour is being investigated the data relating to the warm-up period must be discarded.
- Let O denote the mean value of the retained observations, O_i, after m runs.
- The variance over all observations is calculated as:

$$V = \frac{1}{m-1}\sum_{i=1}^{m}(O_i - O)^2$$

Independent replications and steady-state

For steady-state analysis independent replication is an inefficient way to generate samples, since for each sample point, O_i, k observations, $\{o_{i1}, \ldots, o_{ik}\}$, must be discarded.

Batch means

- In the method of batch means the model is run only once but for a very long period.

Batch means

- In the method of batch means the model is run only once but for a very long period.
- The run is divided into a series of sub-periods of length ℓ, and measures over each sub-run form a single point estimate.

Batch means

- In the method of batch means the model is run only once but for a very long period.
- The run is divided into a series of sub-periods of length ℓ, and measures over each sub-run form a single point estimate.
- If the observations made during the run form a set $\{o_i\}$, the set is partitioned into subsets

$$S_i = \{o_j \mid o_j \text{ observed between } (i-1) \times \ell \text{ and } i \times \ell\}$$

Batch means

- In the method of batch means the model is run only once but for a very long period.
- The run is divided into a series of sub-periods of length ℓ, and measures over each sub-run form a single point estimate.
- If the observations made during the run form a set $\{o_i\}$, the set is partitioned into subsets

$$S_i = \{o_j \mid o_j \text{ observed between } (i-1) \times \ell \text{ and } i \times \ell\}$$

- Each sample point O_i is the mean generated from a subset of observations S_i, and O is the mean generated from the O_i.

Batch means

- In the method of batch means the model is run only once but for a very long period.
- The run is divided into a series of sub-periods of length ℓ, and measures over each sub-run form a single point estimate.
- If the observations made during the run form a set $\{o_i\}$, the set is partitioned into subsets

$$S_i = \{o_j \mid o_j \text{ observed between } (i-1) \times \ell \text{ and } i \times \ell\}$$

- Each sample point O_i is the mean generated from a subset of observations S_i, and O is the mean generated from the O_i.
- Variance is calculated as above.

Batch means and independence

This method is unreliable since the sub-periods are clearly not independent.

Batch means and independence

This method is unreliable since the sub-periods are clearly not independent.

However it has the advantage that only one set of observations $\{o_i \dots o_k\}$ needs to be discarded to overcome the warm-up effects in steady state analysis.

Regeneration

- It is sometimes possible within the run of a simulation model to identify points in the trajectory where the model returns to exactly equivalent states: so-called regeneration points.

Regeneration

- It is sometimes possible within the run of a simulation model to identify points in the trajectory where the model returns to exactly equivalent states: so-called regeneration points.
- Periods between regeneration points are genuinely independent sub-runs, e.g. a queue which empties.

Regeneration

- It is sometimes possible within the run of a simulation model to identify points in the trajectory where the model returns to exactly equivalent states: so-called regeneration points.
- Periods between regeneration points are genuinely independent sub-runs, e.g. a queue which empties.
- The behaviour of the model (queue length, waiting time etc) after a visit to such a state does not depend on the previous history of the model in any way.

Regeneration

- It is sometimes possible within the run of a simulation model to identify points in the trajectory where the model returns to exactly equivalent states: so-called regeneration points.
- Periods between regeneration points are genuinely independent sub-runs, e.g. a queue which empties.
- The behaviour of the model (queue length, waiting time etc) after a visit to such a state does not depend on the previous history of the model in any way.
- The duration between two successive regeneration points is called a regeneration cycle.

Variance computation and regeneration

- The variance computation using regeneration cycles is a bit more complex than that in the method of batch means or the method of independent replications.
- This is because the regeneration cycles are of different lengths, whereas in the other two methods the batches or replications are all of the same length.

Regeneration considered

Unlike the previous two methods, the method of regeneration does not require any transient observations to be removed.

Regeneration considered

Unlike the previous two methods, the method of regeneration does not require any transient observations to be removed.

Unfortunately not all models have easily defined regeneration states, and even when they exist they can be computationally expensive to identify.

Regeneration considered

Unlike the previous two methods, the method of regeneration does not require any transient observations to be removed.

Unfortunately not all models have easily defined regeneration states, and even when they exist they can be computationally expensive to identify.

Another disadvantage is that it is not possible to define the length of a simulation run beforehand.