Random Number and Random Variate Generation

Simulation Coverage to Date

1. Simulation components and simulation mechanics (manual simulation).
2. Data collection, distribution fitting, parameter estimation, goodness of fit tests.
3. Running models and analysis – Steady state vs. terminating analysis, initial conditions, warm-up period, Output analysis.
4. Verification and validation

Simulation fundamentals → Specific implementation (Arena)

Random Number and Random Variate Generation

Simulation Model

Data
Interarrivals
Service times
Breakdowns
Repair times
…

Dist. Fitting

Goodness-of-fit tests

Event driven model
- •Event list
- •Functions
  - Generation of random
    - Interarrivals
    - Service times
    - Breakdowns
    - Repair times
    - …

Simulation replications

Warm-up

Samples of system Performance

x_1
x_2
…
x_n

Output Analysis

Random Number and Random Variate Generation

- How does the computer generate observations from various distributions specified after input analysis?
- There are two main components to the generation of observations from probability distributions.

1.

Random Number and Random Variate Generation

- Random number generation –

- Random Number Generator -

Random Number Generation

- One early method – the midsquare method (von Neumann and Metropolis 1940)
  - Start with a four digit positive integer Z_0.
  - Square Z_0 to get an integer with up to eight digits (append zeros if less than eight).
  - Take the middle four digits as the next four digit integer Z_1.
  - Place a decimal point to the left of Z_1 to form the first "U(0,1)" observation.
  - Repeat

Random Number Generation

- The prior method does not work well.
  - Degenerates to zero.
- What are good methods?
  -

Random Number Generation

- Linear Congruential Generators (LCGs).
- A LCG generates a sequence of integers $Z_1, Z_2, Z_3, \ldots$ using the following recursive formula,
-

Random Number Generation

- Since the mod m operation is used, all Z_i's will be between 0 and m-1.
- To get the "U(0,1)" random observations each Z_i generated is divided by m.

- So are the U_i's really U(0,1) random observations ?

In-class Exercise

- Let m=63, a=22, c=4 and Z_0 =19.
  - Generate the first five "U(0,1)" observations.

Random Number Generation

- How many have heard of the term "random number seed"?
- What is the notation for the seed for a LCG?

- The random number "stream" can be controlled with the seed.
  - Each time the same seed is used, the sequence of U(0,1) observations generated is identical.

Random Number Generation

- Since the “stream” of random numbers generated is reproducible, random number generation procedures are also referred to as _

Random Number Generation

- m, a, and c are the parameters of the random number generator.
  - There can be an infinite number of different implementations of a LCG.
- The values used for m, a, and c determine whether the generator is good or bad.

Random Number Generation

- The stream or sequence of numbers produced by a generator should pass statistical tests for randomness.
  - An outside observer should not be able to tell the difference (statistically) between a stream of pseudo random numbers and an actual random number stream.

Example - Random Number Generation

i	$22*Z_{i-1}+4$	Z_i	U_i	i	$22*Z_{i-1}+4$	Z_i	U_i	i	$22*Z_{i-1}+4$	Z_i	U_i	i	$22*Z_{i-1}+4$	Z_i	U_i
0		19		16	356	41	0.6508	32	840	21	0.3333	48	400	22	0.3492
1	422	44	0.6984	17	906	24	0.3810	33	466	25	0.3968	49	488	47	0.7460
2	972	27	0.4286	18	532	28	0.4444	34	554	50	0.7937	50	1038	30	0.4762
3	598	31	0.4921	19	620	53	0.8413	35	1104	33	0.5238	51	664	34	0.5397
4	686	56	0.8889	20	1170	36	0.5714	36	730	37	0.5873	52	752	59	0.9365
5	1236	39	0.6190	21	796	40	0.6349	37	818	62	0.9841	53	1302	42	0.6667
6	862	43	0.6825	22	884	2	0.0317	38	1368	45	0.7143	54	928	46	0.7302
7	950	5	0.0794	23	48	48	0.7619	39	994	49	0.7778	55	1016	8	0.1270
8	114	51	0.8095	24	1060	52	0.8254	40	1082	11	0.1746	56	180	54	0.8571
9	1126	55	0.8730	25	1148	14	0.2222	41	246	57	0.9048	57	1192	58	0.9206
10	1214	17	0.2698	26	312	60	0.9524	42	1258	61	0.9683	58	1280	20	0.3175
11	378	0	0.0000	27	1324	1	0.0159	43	1346	23	0.3651	59	444	3	0.0476
12	4	4	0.0635	28	26	26	0.4127	44	510	6	0.0952	60	70	7	0.1111
13	92	29	0.4603	29	576	9	0.1429	45	136	10	0.1587	61	158	32	0.5079
14	642	12	0.1905	30	202	13	0.2063	46	224	35	0.5556	62	708	15	0.2381
15	268	16	0.2540	31	290	38	0.6032	47	774	18	0.2857	63	334	19	0.3016

Random Number Generation

- What will happen after the 63^{rd} number is generated?

Random Number Generation

- The example LCG demonstrates cycling in the prior table.
  - Since m=63, it can generate at most 63 numbers before it repeats the same sequence.
- This small random number generator has full period since it generates all possible (m=63) numbers before cycling.
  -

Random Number Generation

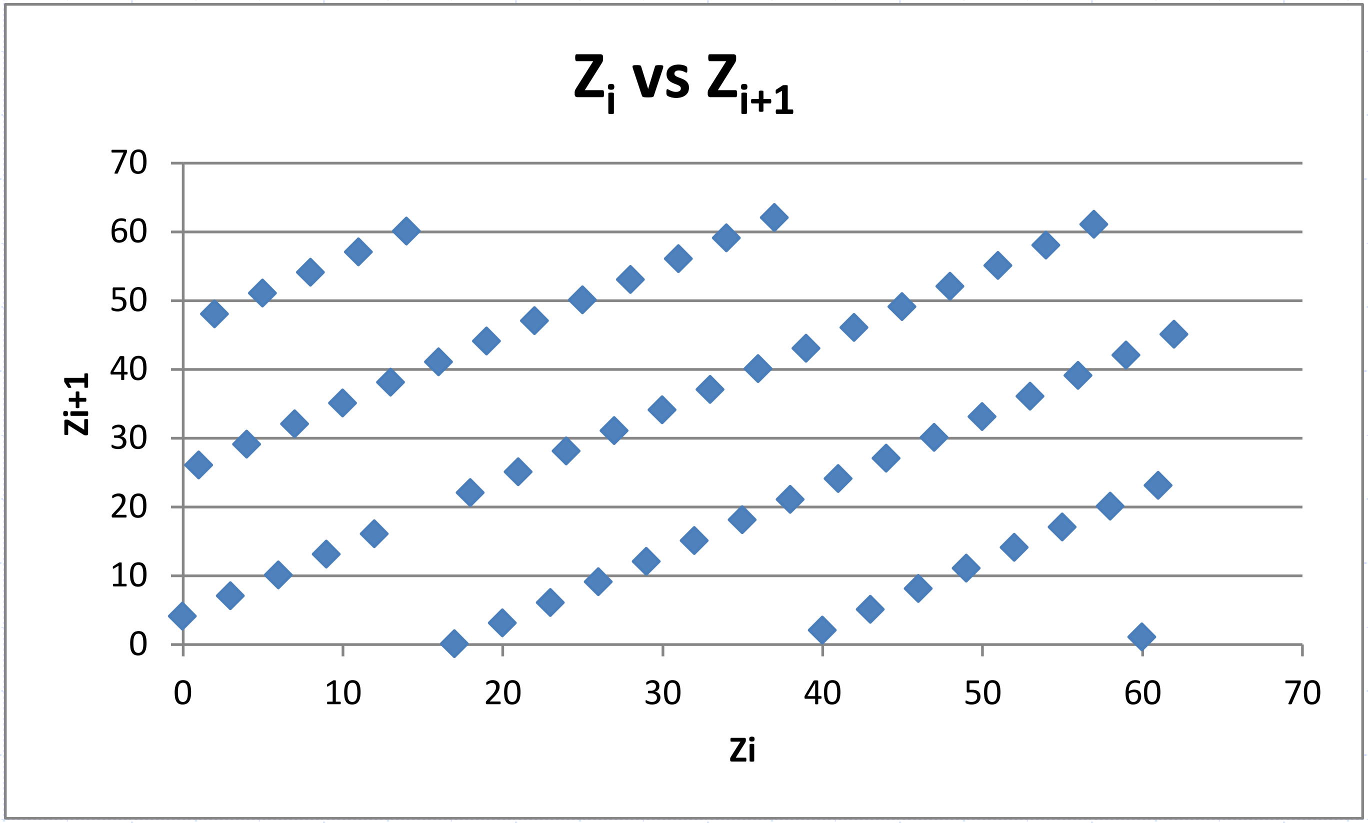
- Theorem (Hull and Dobell 1962)
  - The LCG $Z_i = (aZ_{i-1} + c) \bmod m$ has full period if and only if the following three conditions hold.
    1. The only positive integer that exactly divides both m and c is 1.
    2. If q is a prime number that divides m, then q divides a-1.
    3. If 4 divides m, then 4 divides a-1.

- The parameters of the LCG dictate the period length of the LCG as well as other properties of the numbers generated.

Random Number Generation

- The example (m = 63) generator has full period but bad statistical properties (next slide).
- A good random number generator will have values for *m, a*, and *c* such that full or close to full period is obtained, as well as good statistical properties.
  - Crystal Ball

Random Number Generation



Random Number Generation

- Crystal Ball Demo

Random Number Generation

- Types of LCGs
  - When $c = 0$, the LCG is called a
  - When $c \neq 0$, the LCG is called a
- Most LCGs implemented are multiplicative
  - Can't have full period.
- How is m selected.
  - A large period is desired $\rightarrow$ $m=2^{31}$ (based on a 32 bit word size).
    - In computing, "**word**" is a term for the natural unit of data used by a particular computer design. A word is simply a fixed-sized group of bits that are handled together by the machine.

Random Number Generation

- With m=2^{31} it has been proven that the period can be at most 2^{29} (25% of the values are cycled and gaps may be present).
- m has been selected as the largest prime number less than m=2^{31}, which is 2,147,483,647 = 2^{31} -1.
  - A period of $m-1$ can be guaranteed if the parameter a is primitive element modulo m.
    - The smallest integer k for which a^k-1 is evenly divisible by m is $k = m-1$.

Random Number Generation

- Selections for *a* that are primitive element modulo m and have been in use are:
  - 16,807
  - 630,360,016

Random Number Generation

- Recent (2001) evidence has demonstrated that LCGs with m=2,147,483,647 = 2^{31} -1, and *a*=16,807 or *a*=630,360,016 can show poor statistical properties over small sample sizes.
- Additionally, because of the availability of more computing power, more complex simulations are possible.
  - Cycle lengths of 2^{31} may be too small.
- 64 bit word generators have been developed.

Random Number Generation

- Additionally, other types of random number generators have been developed.
  - More general congruences

$$Z_i = g(Z_{i-1}, Z_{i-2}, \ldots)(\text{mod } m)$$

$$\text{e.g., } g(Z_{i-1}, Z_{i-2}, \ldots) = a_1 Z_{i-1} + a_2 Z_{i-2} + \cdots + a_q Z_{i-q}$$

  - Composite generators
    - Combine outputs from separate random number streams to create a new stream.
  - Other types of generators

Random Number Generation

- Microsoft Excel 2010

U_{1i}, U_{2i}, U_{3i} are the ith random numbers from three separate generators.

U_i = fractional part of $U_{1i} + U_{2i} + U_{3i}$

- Arena implements a random number generation method called a combined multiple recursive generator (a composite generator).

Random Number Generation

$$A_n = (1403580 A_{n-2} - 810728 A_{n-3}) \bmod 4294967087$$

$$B_n = (527612 B_{n-1} - 1370589 B_{n-3}) \bmod 4294944443$$

$$Z_n = (A_n - B_n) \bmod 4294967087$$

$$U_n = \frac{Z_n}{4294967088} \text{ if } Z_n > 0$$

$$U_n = \frac{4294967087}{4294967088} \text{ if } Z_n = 0$$

How many seeds are required?

Random Number Generation

- The constants found in this generator have been selected to give a long cycle length and good statistical properties.
  - Statistical tests of randomness have been passed.
    - Plots of random numbers up to 45 dimensions have been constructed with no patterns.
  - Cycle length = 3.1×10^{57} (too long for cycling to ever occur today).

Random Number Generation

- Controlling random number generation.
  - For a LCG the seed can be specified.
    -
  - Arena – No provision for specifying seeds (six required).
    - The 3.1×10^{57} stream is broken up into 1.8×10^{19} separate streams of length 1.7×10^{38}. Each stream is further separated into substreams (2.3×10^{15} substreams of length 7.6×10^{22}.

Random Number Generation

- Instead of specifying seeds, streams (1 to 1.8×10^{19}) are specified.
  - Default stream is stream 10.
- Within a single random number stream each replication starts at the next substream.

Random Number Generation

- To specify a stream in Arena a distribution must be specified using an Expression.

Random Variate Generation

- Generating “random” or pseudorandom observations from theoretical distributions.
  - Exponential,
  - Normal,
  - Uniform – besides U(0,1),
  - Etc.
- Start with a simple example of a discrete distribution. Let X be a discrete random variable with the following probability mass function

Random Variate Generation

- Next consider the partitioning of the interval [0,1]

- If we generate a U(0,1) observation what is the probability it wall fall in each interval A,B, C?

Random Variate Generation

- The interval presented earlier represents the vertical axis for the cumulative distribution function of X or $F(x)$.

Random Variate Generation

- The distribution function $F(x)$ provides a way to map a U(0,1) observation to specific values according to the probabilities specified in $F(x)$.
-

Random Variate Generation

- Discrete Inverse Transform Algorithm
  1. Generate a U(0,1) observation U.
  2. Find y (among the discrete values of X) such that y is the smallest value with $F(y) \geq U$.
  3. Set $X=y$.

- Verbal interpretation – For what value of x is F(x) first greater than U?

In-class Exercise

- If the mass function for a discrete random variable is

$$P(X = i) = \begin{cases} 0.25 & \text{for } i = 1 \\ 0.25 & \text{for } i = 2 \\ 0.4 & \text{for } i = 3 \\ 0.1 & \text{for } i = 4 \end{cases}$$

- Generate the first four observations of this random variable if the first four U(0,1) observations are 0.35, 0.87, 0.28, 0.36.

Random Variate Generation

- The discrete inverse transform algorithm is general and applies to any discrete random variable with a finite number of possible outcomes.
- Other non-finite discrete distributions have properties which call for other methods.

Random Variate Generation

- Examples (inverse transform)
  - Bernoulli random variables
    - p = probability of success.
    1. Generate U~U(0,1).
    2. If U ≤ p set X=1, else X=0.

  - Discrete Uniform(i,j) random variables
    1. Generate U~U(0,1).
    2. Set X = $i + \lfloor (j - i + 1)U \rfloor$

Random Variate Generation

- Demo in Excel

Random Variate Generation

- Poisson random variable with mean λ (infinite number of possible outcomes)

$$1.\ \text{Let } a = e^{-\lambda},\ b = 1,\ \text{set } i = 0.$$

$$2.\ \text{Generate } U_{i+1} \sim U(0,1),$$

$$\text{set } b = b * U_{i+1}$$

$$\text{if } b < a \ \text{ set } n = i \ \text{ and return}$$
$$\text{else go to 3.}$$

$$3.\ \text{Set } i = i + 1,\ \text{go to 2.}$$

- This is a method called acceptance-rejection.

Random Variate Generation

- Continuous Random Variables
  - The inverse transform method is also used for continuous random variables.
- Inverse transform algorithm – verbal.
  - For a continuous random variable X with distribution function F, first generate $U \sim U(0,1)$ and then find the value of X that gives $F(X)=U$.

Random Variate Generation

- Finding X that gives F(X)=U.
  - The function F can be inverted to obtain F^{-1} which will be a function of U.
  - If u=F(x), then $x=F^{-1}(u)$.

Random Variate Generation

- Diagram

Finding “F Inverse”

In-class Exercise

If $X \sim$ exponential with mean $\dfrac{1}{\lambda}$ then $F(x) = 1 - e^{-\lambda x}$ for $x \geq 0$, and 0 otherwise.

What is $F^{-1}(u)$?

Random Variate Generation

- General continuous inverse transform algorithm
  1. Generate U~U(0,1).
  2. Set $X=F^{-1}(U)$.

Examples

Random Variate Generation

- Why does the algorithm work?

In-class Exercise

- For U(0,1) observations 0.142, 0.432, 0.550, and 0.712 generate observations from an exponential distribution with a mean= 10, and observations from a U(50,100) distribution.

Random Variate Generation

- Not all distribution functions have closed form distribution functions (e.g., normal distribution), and not all distribution functions can be inverted to get $F^{-1}(U)$ in closed form.
- Other methods of random variate generation.
  -

Random Variate Generation

- Composition – When the distribution F can be expressed as a convex combination of other distribution functions $F_1, F_2, \ldots$ and it is straightforward to generate observations from $F_1, F_2, \ldots$

- Algorithm
  - Generate a positive integer j with probability p_j.
  - Return X with distribution F_j.

Random Variate Generation

- Example.

Random Variate Generation

- Convolution – Can be used when the desired random variable can be expressed as the sum of other independent random variables.
- Example
  - $X \sim m\text{-}Erlang$ with mean μ.
  - Then $X = Y_1 + Y_2 + \ldots + Y_m$
  - Where the Y_i are independent, identically distributed exponential random variables with mean μ/m.

Random Variate Generation

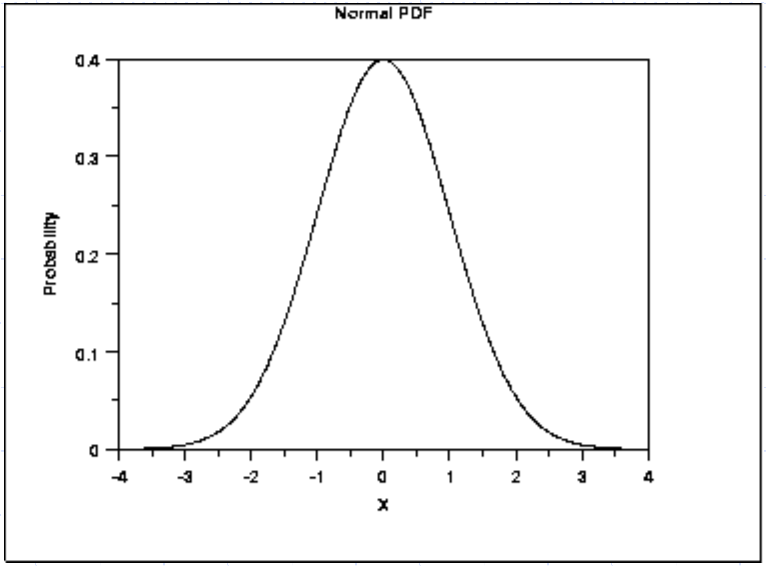
- *m-Erlang* algorithm
  - Generate $U_1, U_2, \ldots, U_m$ as independent $U(0,1)$ random variates.
  - Set

$$X = \frac{-\mu}{m} \ln\left(\prod_{i=1}^{m} U_i\right)$$

Random Variate Generation

- Why is this correct?

Normal Random Variates



- Cannot use the inverse transform algorithm for the standard normal distribution ($\mu = 0$, $\sigma^2 = 1$).
- No formula for F(x)
- Only need to generate Normal(0,1)

Box-Muller Method

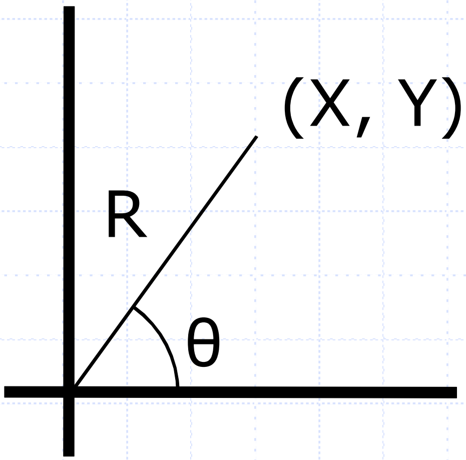
- U_1, U_2 are independent U(0,1) random variables

$$Z_1 = \sqrt{-2\ln U_1}\cos\left(2\pi U_2\right)$$

$$Z_2 = \sqrt{-2\ln U_1}\sin\left(2\pi U_2\right)$$

Why Does it Work?

- X, Y independent standard normal random variables
- Consider the point (X, Y) in polar coordinates



$$R^2 = X^2 + Y^2$$

$$\tan\theta = \frac{Y}{X}$$

Why Does it Work?

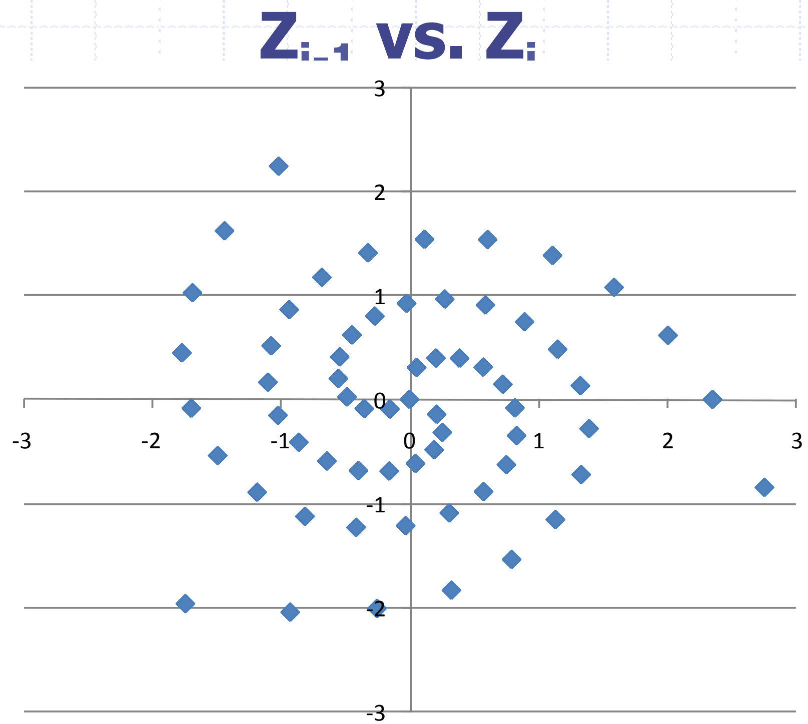
Notes on Box-Muller Method

- Faster methods exist
  - Generally more complicated

- Caution when using with a single LCG
  - U_1, U_2 no longer independent (if adjacent)
  - Solution: use multiple LCG's or another form of generator, e.g. composite

Box-Muller LCG Example

U_{i-1} and U_i from the same LCG



Variance Reduction

- Simulation models are used to estimate system performance measures.
- System performance = f(system parameters, system design).
  - Simulation is an approximation of the function f.
- Since there are random system components each run of the simulation produces an observation of the value of f.
- To get more precision about the value of f, more simulation runs can be performed.

Variance Reduction

- If there is high system variance then many runs may be required to get the necessary precision.
  - Runs times may be too long.
  - Simulation cannot respond fast enough.
-

Variance Reduction

- Can you reduce the variance in the system?
- However, since there is control over the pseudo random number generators (or streams in Arena), it may be possible to use this control to reduce the variance in certain situations.

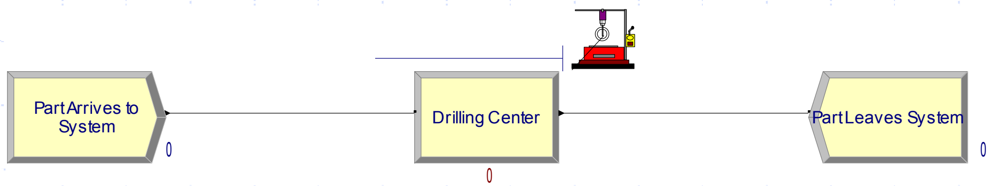
Variance Reduction

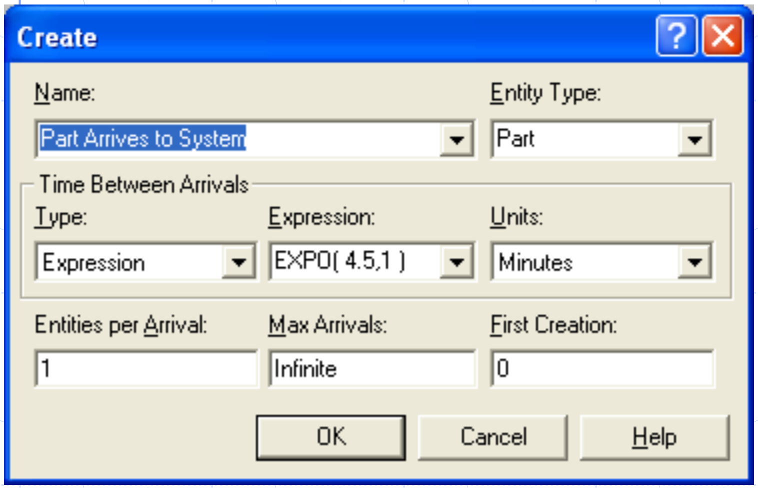
- Comparison of two systems.
  - Interested in the difference in the performance of two systems.
  - Intuition
    - The differences in two variations of a system are due to
      1. Changes in the system and or system parameters.
      2. The realization of random components in the system (e.g., failure times, inspection results,…)
    -

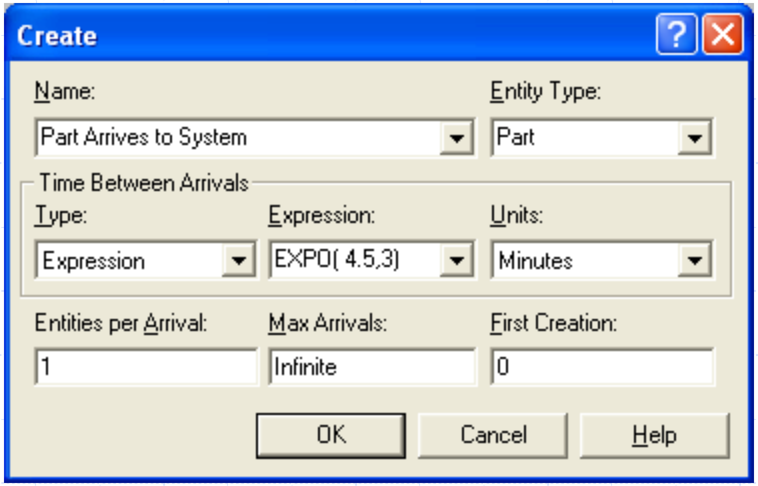
Variance Reduction

- Try to minimize the differences observed between two systems caused by differences in random components (e.g., arrival times, process times, etc.).
  -

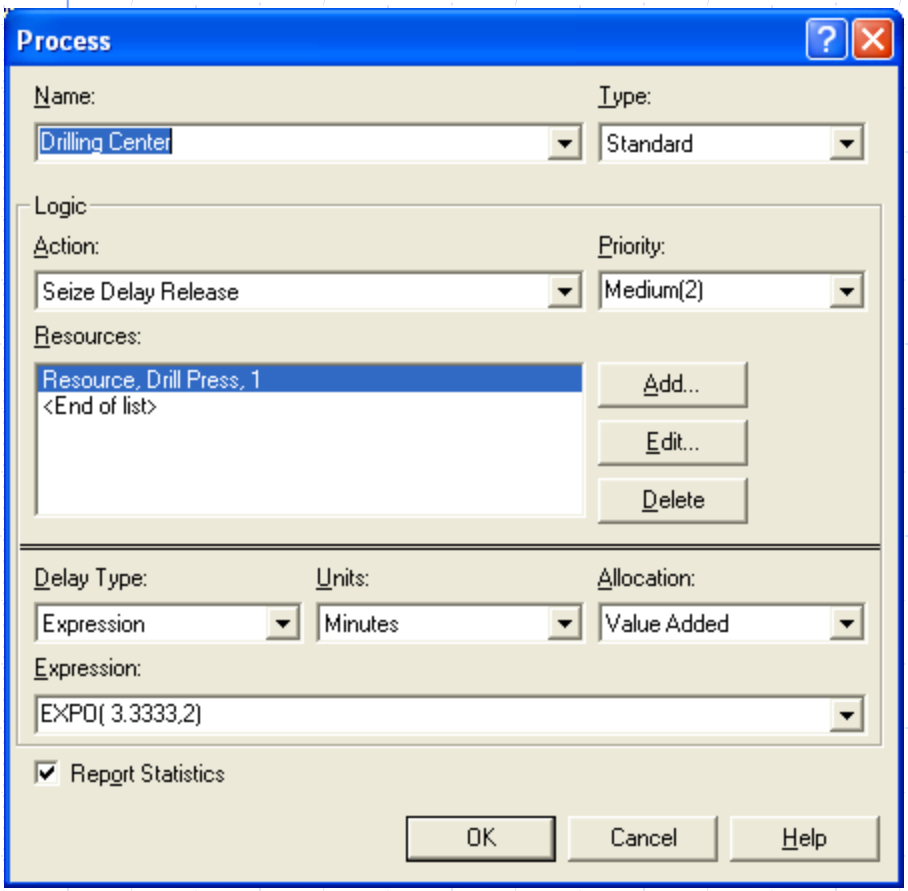
Arena – With no CRN

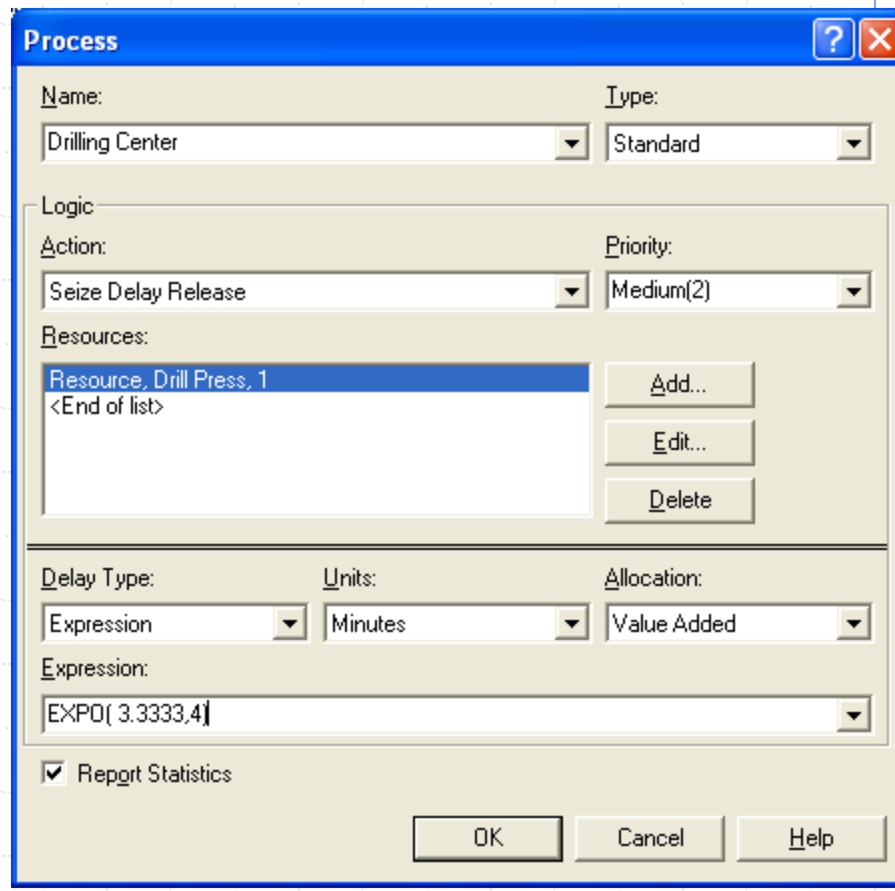




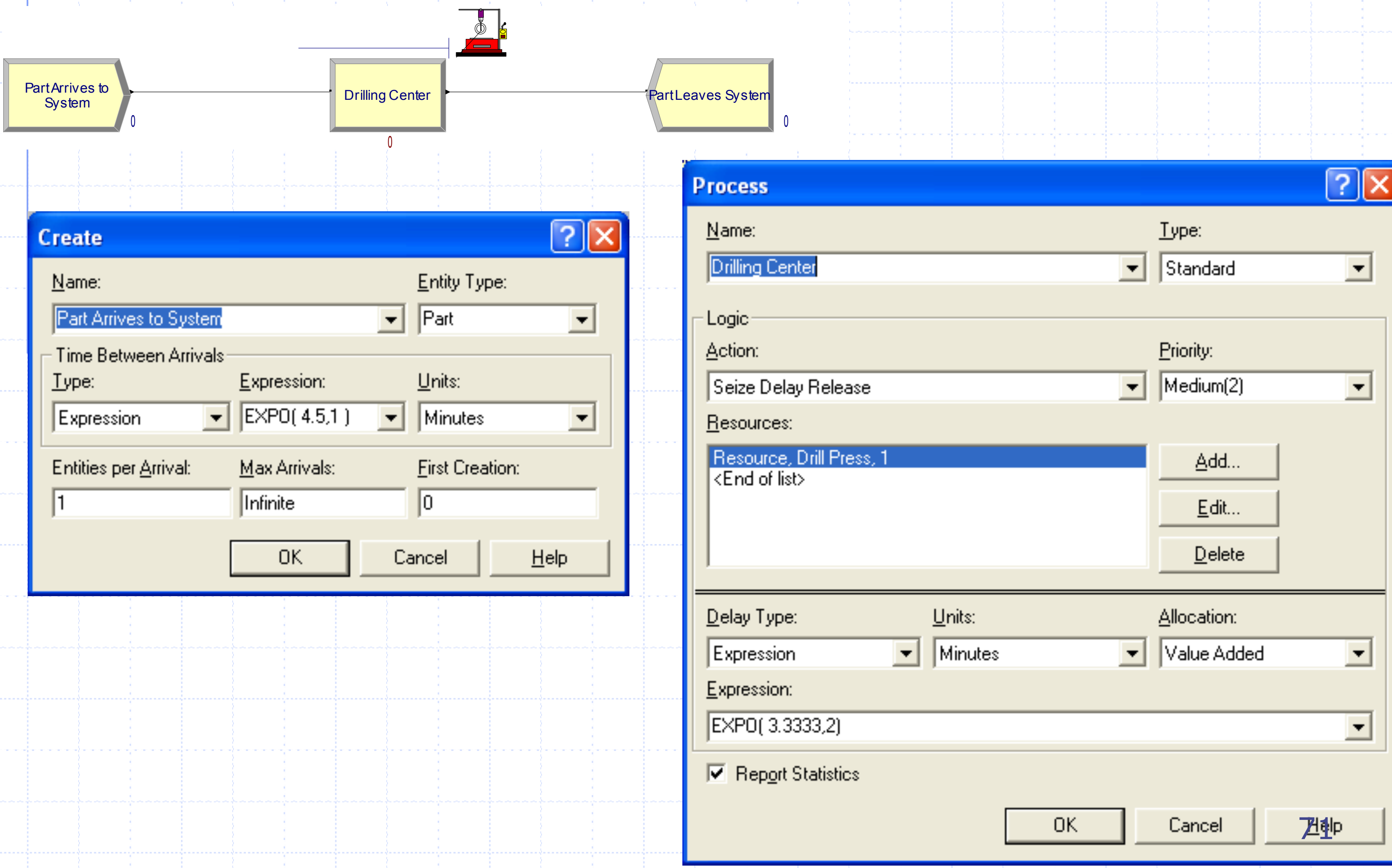


Arena – With no CRN





Arena - With CRN



Arena - Results

No CRN (80 hrs. per rep.)			
Rep.	Mean = 5	Mean = 4.5	Difference
1	10.55593259	19.15882425	8.602891651
2	11.62190972	10.52054723	-1.101362492
3	10.73667873	11.66566449	0.928985759
4	9.441779109	9.447215439	0.00543633
5	11.13851111	11.98255637	0.844045264
6	10.12244579	13.31500819	3.192562394
7	11.46097002	9.821655377	-1.639314645
8	9.666154875	16.89704156	7.230886687
9	9.678278443	26.04126443	16.36298598
10	8.893112092	12.55677097	3.663658876
11	13.89013848	13.37045172	-0.519686758
12	11.5716041	13.75993899	2.18833489
13	9.544192578	12.38687381	2.842681234
14	11.11666579	9.011359994	-2.105305799
15	8.859603189	10.53319565	1.673592459
16	9.271500665	9.672631262	0.401130597
17	8.68341031	9.642327162	0.958916852
18	8.714678262	10.10640835	1.391730089
19	11.22636828	16.00750627	4.781137993
20	9.328160211	13.07080157	3.742641354
Avg	10.276	12.948	2.672
StdDev	1.326	4.091	4.234

With CRN (80 hrs. per rep.)			
Rep.	Mean = 5	Mean = 4.5	Difference
1	10.55593259	13.44001444	2.884081849
2	11.62190972	14.50311678	2.88120706
3	10.73667873	14.20560421	3.468925471
4	9.441779109	11.32154226	1.879763151
5	11.13851111	13.64415642	2.505645311
6	10.12244579	12.79783734	2.675391542
7	11.46097002	16.03953701	4.57856699
8	9.666154875	11.39411414	1.727959263
9	9.678278443	11.89496922	2.216690779
10	8.893112092	12.87069329	3.977581201
11	13.89013848	17.72575688	3.835618397
12	11.5716041	14.83878646	3.267182361
13	9.544192578	12.0015145	2.457321924
14	11.11666579	14.76231143	3.645645634
15	8.859603189	11.32017699	2.460573801
16	9.271500665	11.1423295	1.870828831
17	8.68341031	10.42226041	1.738850098
18	8.714678262	10.52733463	1.812656364
19	11.22636828	14.10493336	2.878565083
20	9.328160211	14.19855829	4.87039808
Avg	10.276	13.158	2.882
StdDev	1.326	1.936	0.941

Statistical Rationale for CRN

Let

X_{1j} = Output (e.g., avg TIS) from the jth indepedent simulation replication from system 1.

X_{2j} = Output from the jth indepedent simulation replication from system 2.

The objective is to estimate

$$d = \mu_1 - \mu_2, \text{ where } \mu_1 = E(X_1), \mu_2 = E(X_2).$$

Statistical Rationale for CRN

If n replications of each (system 1 and 2) simulation are performed, an estimate of d is

$$\bar{d}(n) = \frac{\sum_{j=1}^{n} d_j}{n} = \frac{\sum_{j=1}^{n} (X_{1j} - X_{2j})}{n}$$

Since the paired runs may be correlated but the runs are independent

$$Var[\bar{d}(n)] = Var\left[\sum_{j=1}^{n} d_j / n\right] = \frac{n * Var(d_j)}{n^2} = \frac{Var(X_{1j} - X_{2j})}{n}$$

$$= \frac{Var(X_{1j}) + Var(X_{2j}) - 2 * Cov(X_{1j}, X_{2j})}{n}$$

Statistical Rationale for CRN

If $Cov(X_{1j}, X_{2j}) > 0$ then $Var[\bar{d}]$ is reduced.

CRN attempts to introduce this positive correlation.

It is possible that CRN can increase variance if negative correlations are induced.

To check let:

$s_d^2(n) =$ Sample variance of the d_i's.
$s_{X_1}^2(n) =$ Sample variance of the X_{1j}'s.
$s_{X_2}^2(n) =$ Sample variance of the X_{2j}'s.

If $s_d^2(n) < s_{X_1}^2(n) + s_{X_2}^2(n)$ then CRN is working as intended.