Appendix A

INTRODUCTION TO STATISTICS OF SIMULATION

This appendix covers basic probability concepts and statistical methods for simulation. Software processes are stochastic like other real-life processes. The mere presence of people alone guarantees a degree of uncertainty. Treating software processes as deterministic is a gross approximation. In order to build representative simulation models, the random effects present in a system must be simulated. This appendix will show methods to deal with random effects in the inputs to system dynamics models as well as analysis of stochastic simulation output. The modeling principles described up to this point have mostly treated simulation outputs as deterministic point estimates (except for a Monte Carlo example in Chapter 2).

However, a point estimate is only one of an infinite number of possibilities; hence, the probability of achievement is zero. One should evaluate the entire uncertainty range. The variance of outputs from multiple runs is used to quantitatively assess risk.

The stochastic nature of a system is thus an essential feature to consider. Bounds on the uncertainty and the implications of potential outcomes must be understood and evaluated. Analytical models have restrictive constraints on the number and kind of random variables that can be included, whereas simulation provides a flexible and useful mechanism for capturing uncertainty related to complex systems.

A simulation can be deterministic, stochastic, or mixed. In the deterministic case, input parameters are specified as single values. Stochastic modeling recognizes the inherent uncertainty in many parameters and relationships. Rather than using point estimates, stochastic variables are random numbers drawn from a specified probability distribution. Mixed modeling employs both deterministic and stochastic parameters.

It can be risky to draw conclusions based on the results of a single simulation run. One should not be blinded by deterministic point estimates. A point estimate may give a first-cut approximation or an initial handle on things, but stochastic analysis is much preferred. In a purely deterministic model, only one simulation run is needed for a given set of parameters, but with stochastic or mixed modeling the result variables differ from one run to another because the random numbers drawn differ from run to run. In this case, the result variables are best analyzed statistically across a batch of simulation runs (see the section on Monte Carlo analysis).

This appendix describes probability distributions and includes methods for analyzing input data, generating random variates, and analyzing the output of simulation models. More comprehensive treatments of these topics can be found in [Law, Kelton 1991] and [Khoshnevis 1992] (the latter is oriented toward discrete systems). It should be noted that many of the methods for discrete system analysis will not apply to continuous modeling. Autocorrelation from queuing processes and output data dependency are not problems in system dynamics. For example, analysis of queue waiting times via instrumentation of entities is not performed in traditional system dynamics since entity flow is aggregated together. Information on individual entities is generally not available with classic system dynamics simulation tools, though some tools allow limited attribute tracking.

[Sterman 2000] is another source for system dynamics model verification and validation techniques. Chapter 2 in this book already covered much of that same material, but [Sterman 2000] includes more procedures and tools with examples of statistical methods that can be applied to continuous models.

A.1 RISK ANALYSIS AND PROBABILITY

Risk means uncertainty that can be described with a probability distribution. Accordingly, risk analysis implies a study to determine the outcomes of decisions along with their probabilities. For example, answering the question "What is the likelihood of completing a software project within 10 calendar months?" is a risk analysis. A decision maker is often concerned with exploring options and evaluating the resultant probabilities that a project can be carried out with the time and money available. The risks of failure may differ between alternatives and should be estimated as part of the analysis.

In the context of software process management, risk analysis deals with identification of adverse impacts of process risks and determination of their respective probabilities. In decision making under uncertainty, risk mitigation aims to minimize the failure to achieve a desired result, particularly when that result is influenced by factors not entirely under the decision maker's control.

With stochastic analysis, the range of possible inputs and outputs is analyzed. Few parameters are known with certainty and few things in the software process are truly deterministic, so probability distributions are used to assess the effects of uncertainties. By looking at the range of possible results, risk analysis can help answer the question "how much is enough?" in terms of balancing process activities. Examples are shown in subsequent sections.

In probabilistic risk analysis, process model input parameters are specified as distributions. Corresponding output distributions reflect the stochastic ranges of the input parameters. Confidence level risk charts are based on the resulting output distribution. A point on the output distribution corresponding to a desired confidence level specifies the risk tolerance.

Confidence level refers to the probability of not exceeding a given value (e.g., a conservative 90% confidence level cost means a 10% chance of exceeding the cost based on the probabilistic inputs). A point estimate should always be reported with its associated confidence level. In lieu of any other information, the default assumption should be that an estimate refers to the 50% confidence level. This means that half the time the actual result will be less and half the time it will be greater.

One common method of probabilistic estimation involves inputting fixed probability distributions, and is described in the next section. The Monte Carlo simulation technique described later is another method used to analyze stochastic systems. Monte Carlo analysis is a "game of chance" technique used to solve many types of problems by applying random sampling methods. Risk analysis using Monte-Carlo simulation takes random samples from input probability distributions and computes corresponding output probability distributions. It estimates the likely range of outcomes by running a model a large number of times.

There are practical uses of the results of a probabilistic estimate. In terms of estimating costs, you may want to budget contingency funds to cover a comfortable part of the range. For fixed-price jobs, this can be especially important. For setting expectations, one should express size/cost/schedule estimates in terms of ranges rather than fixed numbers (point estimates). Stakeholders are better served by discussing the project in terms of its probability of completion (e.g., there is an 80% chance that effort will be less than 200 person-months).

A.2 PROBABILITY DISTRIBUTIONS

Probability is a quantitative measure of the chance or likelihood of an event occurring. The event may be the delivering of software by a certain date, keeping software costs within a given budget, software having a specified reliability, or having an earthquake on a given day.

A probability distribution is an arrangement of data that shows the frequency of measurements versus measurement values (also called a frequency distribution). See Figure A.1 for a general probability distribution function. The graph is also known as the probability density function $f(x)$ of the random variable x_i. The total area under a probability distribution function (PDF) is unity.

Distributions may be continuous (e.g., the height of people or the time of software development) or they can be discrete (e.g., the number of software defects). Intervals are sometimes used on the abscissa to show the number of measurements within discrete bins, and this bar chart depiction is called a histogram. For discrete functions,

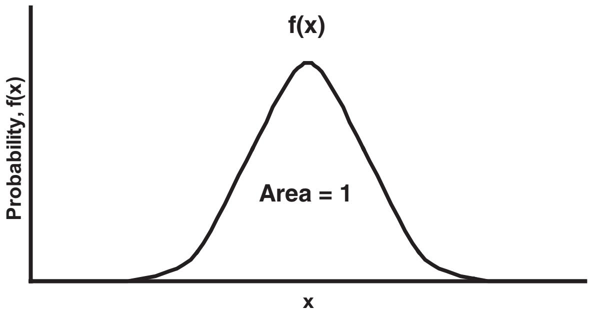


Figure A.1. Probability distribution function.

$F(X)$ is the summation of $f(x)$ and holds the same properties. See Figure A.2 for an example discrete PDF. Generally, we can use continuous probability distributions to also model discrete quantities in the process modeling context, so this section will focus on continuous distributions.

The integral or cumulative form of the PDF is called the cumulative distribution function (CDF). It is also sometimes called a probability distribution function, so we will use the term CDF to distinguish it. The CDF is defined as $F(X)$, the integral of $f(x)$:

$$F(X) = \int f(x)\, dx$$

The CDF has the following properties:

- $0 \leqq F(x) \leqq 1$ for all x
- $F(x)$ is nondecreasing

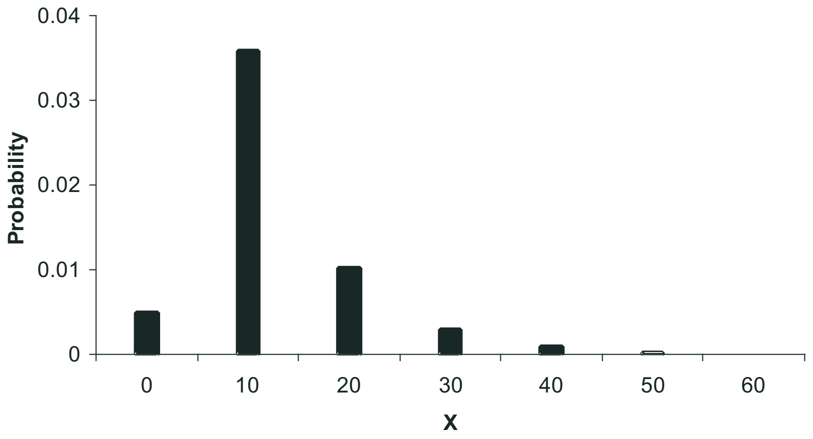


Figure A.2. Discrete probability function.

The CDF gives the probability that a random value x_i is less than or equal to x. This probability is expressed as

$$P(x_i < x)$$

The final value of a CDF is always one, since the entire area under the PDF is also one. Figure A.3 shows a general cumulative distribution corresponding to the PDF in Figure A.1. The PDF is the first derivative of the CDF, since the CDF is the integral of the probability distribution function $f(x)$. A common application of the cumulative distribution is to generate random variates for the Monte Carlo method, and is described in the next major section.

A.2.1 Interpreting Probability Distributions

Recall that the total area under a probability density function is unity. The area under a portion of the probability density function is the probability of a random measurement x_i lying between the two corresonding value x and $x + dx$. In a symmetric normal distribution, 50% of the values lie below the mean and 50% above.

For example, the shaded area in Figure A.4 corresponds to the probability of size being between 40 and 45. It equals the difference of cumulative probabilities evaluated at 45 and 40 shown on the cumulative distribution in Figure A.5, or $F(45) - F(40) = 0.309 - 0.159 = 0.150$. Thus, there is a 15% chance of the size lying between 40 and 45.

The cumulative form of a resulting output distribution can be used as a confidence level chart. Confidence level refers to the probability of not exceeding a given value. See Figure A.6 for a PDF and corresponding CDF that can be used to determine the confidence level as the cumulative probability for a given value.

A sample confidence level chart in the form of a transposed cumulative effort distribution is shown in Figure A.7. The figure represents the results of running many simulation runs, and can be used to assess project cost risk. For example, there is an 80% chance that the project will take 30,000 person-months or less.

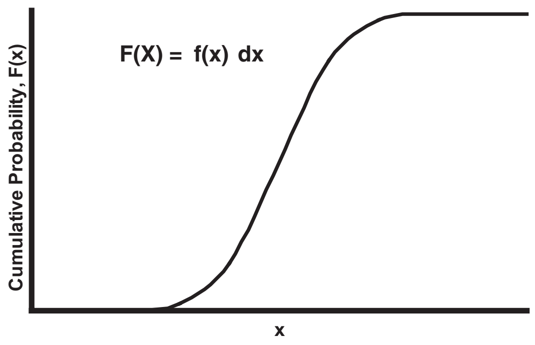


Figure A.3. Cumulative distribution function.

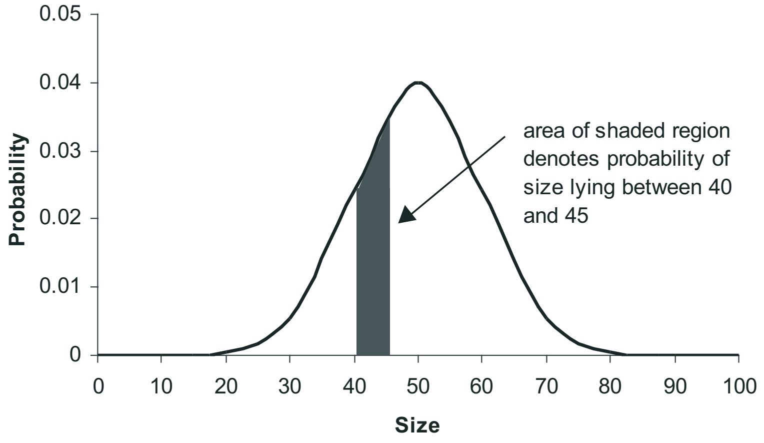


Figure A.4. Interpreting the probability region on a normal probability distribution.

A.2.2 Measures of Location, Variability, and Symmetry

Data and their probability distributions can be summarized in terms of their measures of location (usually a measure of central tendency), variability (referring to how the measures are bunched together or scattered apart), and symmetry. A measure of central tendency is not adequate to give a complete picture of a distribution. Following are the typical measures used:

Mean or average—the measure found by adding together all the measurements of a distribution and dividing by the total number of measurements

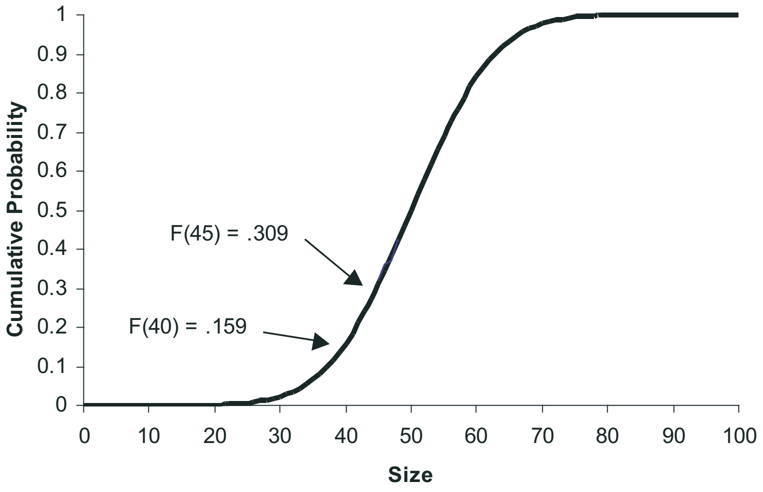


Figure A.5. Calculating probability from cumulative distribution.

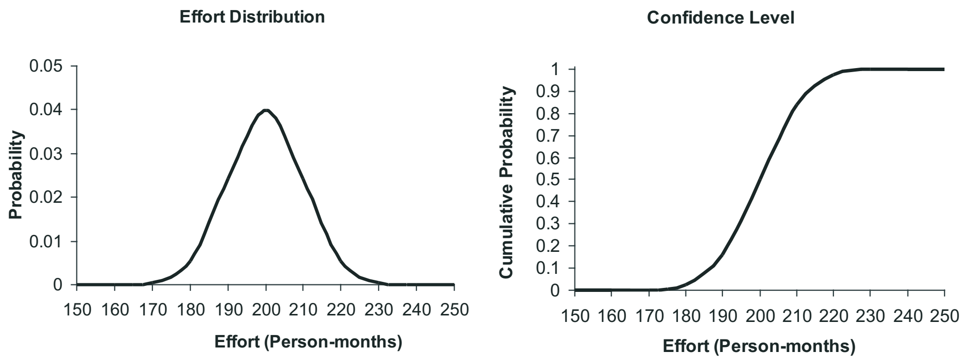


Figure A.6. Sample output distributions.

Median—the middle measurement in a distribution when the measures are arranged from largest to smallest

Mode—the measurement that occurs most often in a distribution

Range—the difference between the lowest and highest measurements

Deviation—the difference between an individual measurement and the mean of a distribution

Variance—the mean of the squared deviations in a distribution

Standard deviation—the square root of the variance; or the square root of the mean of the sum of the squared deviations in a distribution

Skewness—describes how a distribution is shaped in terms of symmetry

Standard deviation is more commonly used to describe the scatter than variance because it uses the same units as the measurements. Standard deviations give a good indication of how wide a distribution is. A small standard deviation indicates a much

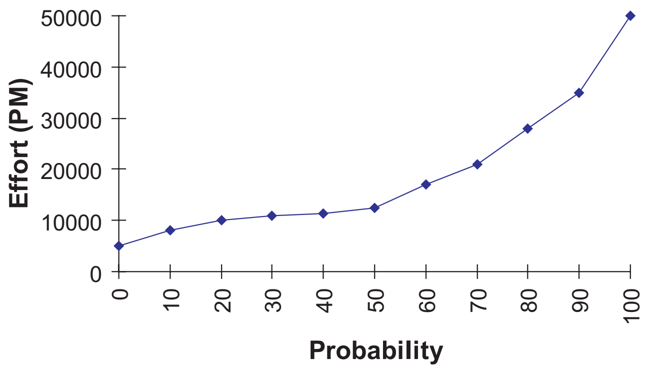


Figure A.7. Confidence level chart using transposed cumulative probability distribution.

narrower distribution than a large deviation. Thus, a statement of just the mean and standard deviation is enough to give a rough picture of a whole distribution.

In a normal distribution, for example, about 68% of measurements lie within one standard deviation on either side of the mean. About 95% lie within two standard deviations, and about 99% within three standard deviations.

Formulas for calculating these distribution parameters are provided in more detailed statistics books. A good reference for the formulas in the context of simulation modeling is [Law, Kelton 1991]. Normally, a modeler does not need to know the formulas since most simulation packages have the distributions predefined. A spreadsheet is another handy adjunct tool to create custom distributions outside of a simulation tool, and the data can be easily imported into a simulation.

A.2.3 Useful Probability Distributions

There are a very large number of probability distributions used in different fields. We will overview some of the simpler distributions that can be applied to software process modeling.

A.2.3.1 Uniform

A uniform distribution is one in which there are an equal number of measures in each interval. It represents an equal probability between its endpoints, as shown in Figure A.8. Any value in the range is equally likely. The uniform distribution $U(0,1)$ is essential for generating random values for all other distributions. It can be used as an initial model for a quantity that is felt to be randomly varying between a and b but about which little else is known.

A.2.3.2 Triangular

The triangular distribution is used when the minimum, maximum, and most likely values of a random variable are known but no other information. A triangular distribution

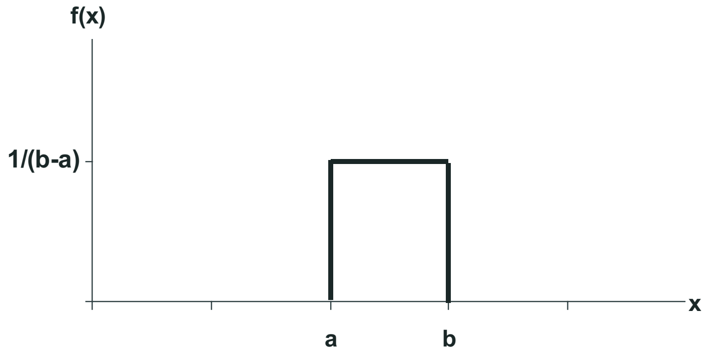


Figure A.8. Uniform distribution.

is often used as a rough model in the absence of data. It accounts for a tapered distribution with the values being centered around the *c* location parameter. The limiting cases where $c = b$ and $c = a$ are called right-triangular and left-triangular distributions respectively.

A.2.3.3 Normal

A normal distribution is symmetric around the mean and bell shaped (it is also called a bell curve). The most measures occur in the intervals closest to the center of the distribution and fewer in the intervals farther from the center. Figure A.10 shows sample normal probability distributions for the random variable size with different standard deviations (σ). All distributions in the figure have a mean of 50.

Figure A.11 shows the corresponding CDF for the normal distribution in Figure A.10.

A.2.3.4 PERT

The PERT probability distribution used in many cost estimation and scheduling tools is a form of a beta distribution. It is a rounded version of the triangular distribution that can be skewed or resemble a normal distribution. It is specified by three parameters: a minimum, a most likely, and a maximum value. Typically, the minimum and maximum represent 5% and 95% cumulative probabilities, respectively.

A.2.3.5 Lognormal

A skewed distribution is one in which the measurements cluster around a certain value, but that value is not in the center of the distribution. One example is the lognormal distribution, in which $\ln(x)$ is normally distributed. The distribution mean is the mean of $\ln(x)$, and the standard deviation is the standard deviation of $\ln(x)$. Figure A.13 shows three sample lognormal distributions in which the mean of $\ln(x)$ is 0 and standard deviations and 0.5, 1, and 1.5.

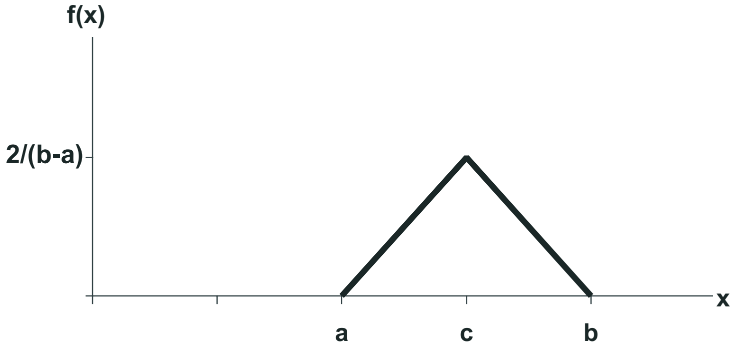


Figure A.9. Triangular distribution.

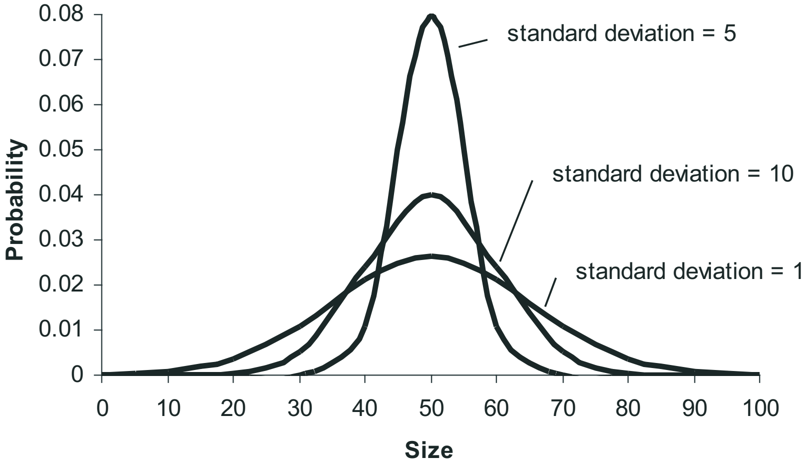


Figure A.10. Normal probability distributions (mean = 50).

The lognormal function can take on shapes similar to the gamma function or the Weibull distribution (the Rayleigh curve is actually a special case of the Weibull distribution). The density takes on shapes similar to gamma (a,b) and Weibull (a,b) for densities for $a > 1$, but can have a large spike close to $x = 0$.

One drawback of a lognormal function is that it frequently has an extremely long tail. Hence, we resort to a truncated lognormal function to bound the distribution to more realistic end values. The previous distribution is shown truncated in Figure A.14. This distribution is often a good one to model the overall size or complexity of a software project, as one tends to underestimate the maximum possible values. Studies have shown that a lognormal distribution describes the ratio of actual effort to estimates from completed projects.

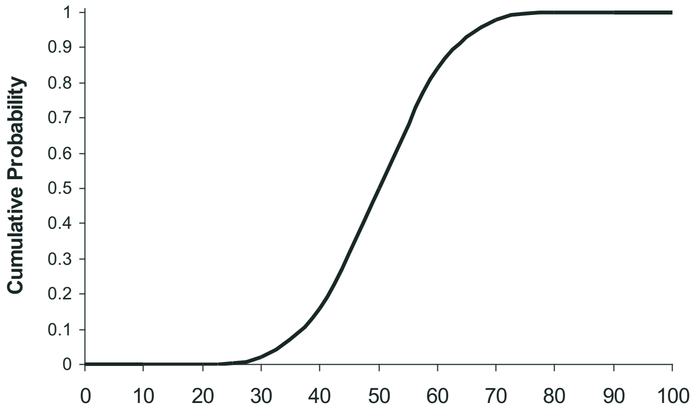


Figure A.11. Cumulative normal probability distribution.

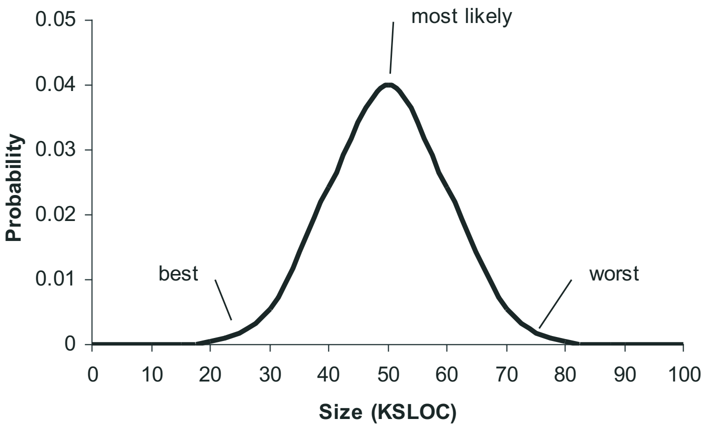


Figure A.12. PERT distribution.

A.2.3.6 Gamma

Another skewed distribution is the gamma function. It is shown in Figure A.15 with varying values of the beta parameter while holding alpha constant. It can be used to model the time to complete a task, or it can also be used in reliability studies to model the time to failure of a software component or system. The gamma function, the log-

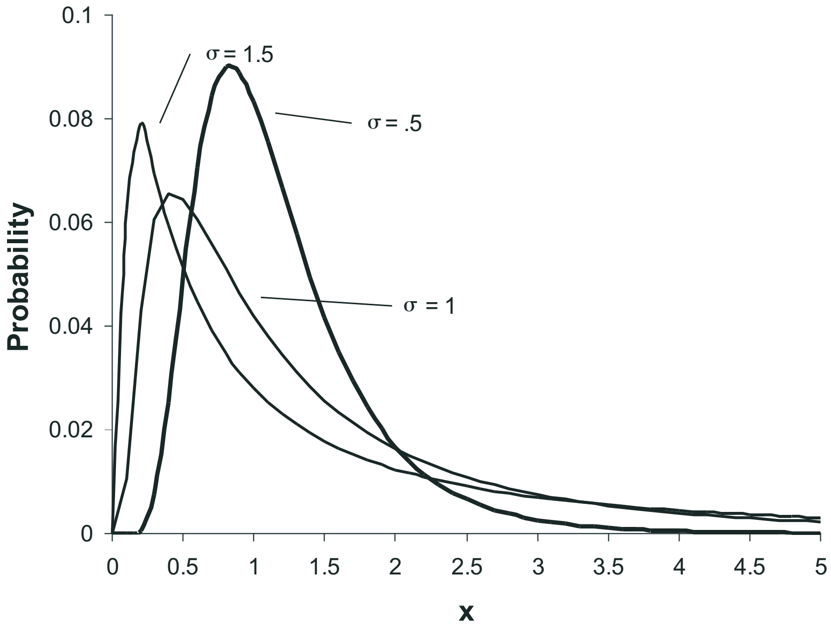


Figure A.13. Lognormal probability distribution.

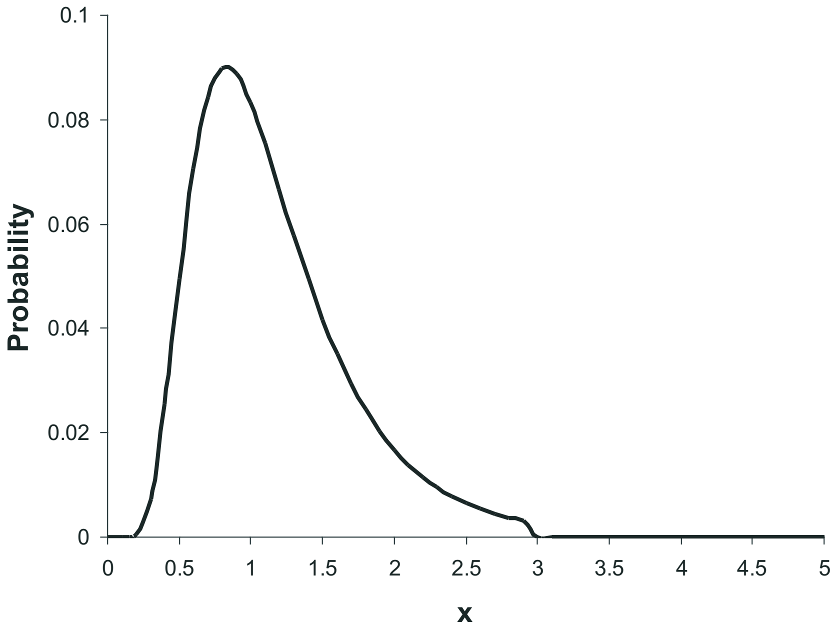


Figure A.14. Truncated lognormal probability distribution.

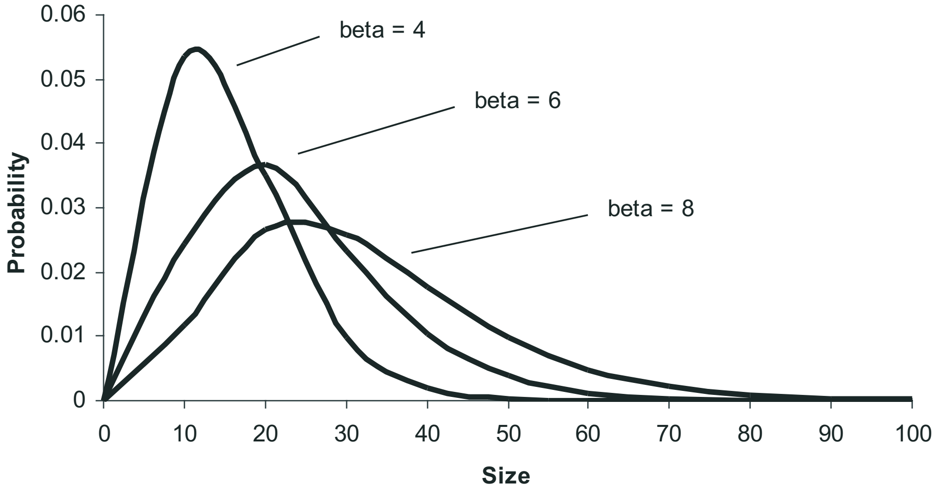


Figure A.15. Gamma probability distribution (alpha = 4).

normal distribution, and the Weibull distribution (Rayleigh curve) can take on very similar shapes.

A.2.3.7 Empirical Distributions

When actual observed data is available, an empirical distribution can be derived from it instead of resorting to a theoretical distribution. Sometimes, a theoretical distribution will not adequately fit the data, and often an empirical distribution will be more realistic since it is predicated on real data (assuming there is enough data). For example, some data might exhibit a bimodal distribution and the best way to represent it is building up your own empirical distribution.

A.2.3.8 Summary of Probability Distributions

A summary of the major characteristics of probability distributions is shown in Table A.1.

What are the best probability distributions to use for software processes? See Table A.2 for basic distributions that can be used in software process modeling. The uniform distribution is a simple one to start out with, and is relevant when there is an equal probability of a variable within a given range or no better information is known. Next in sophistication is the triangular distribution. It does not have to be symmetric, and some skewness can be modeled with it.

Table A.1. Probability distribution summary

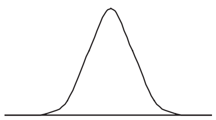
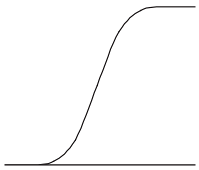
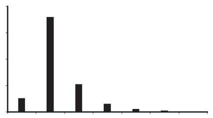
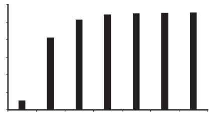
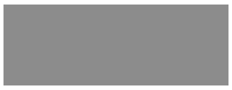
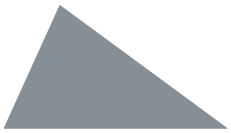
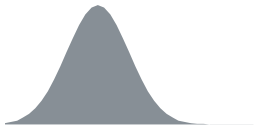
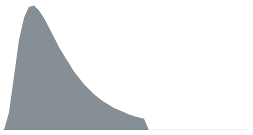
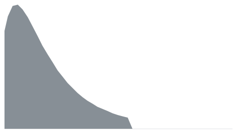
	Probability Distribution		Cumulative Probability Distribution	
	Function	Graph	Function	Graph
Continuous	$P(x < x_i < x + dx) = f(x)$	frequency vs. measurement value	$P(x_i \leq x) = F(X)$	Cumulative percentage
	$f(x)$ is the first derivative of $F(X)$		$F(X) = \int f(x)dx$	
Discrete	$P(x < x_i, x + \Delta x) = f(x)$	frequency vs. measurement value	$P(x_i \leq x) = F(X)$	Cumulative percentage
	$f(x)$ is a term of $F(X)$		$F(X) = \Sigma f(x)$	

Table A.2. Basic distributions for modeling software processes

Distribution	Shape	Typical Applications
Uniform		• Used in the absence of distribution data • Used to represent equally likely outcomes (e.g., decision choices) • Software cost drivers
Triangle		• A rough model to use when the minimum, most likely, and maximum are known but no other distribution shape data • Software cost drivers, including size
Normal (or PERT)		• Personnel factors representing a normal spread of capability • Size, effort, complexity, and other project characteristics • PERT is available in many software project estimation or modeling tools; it can be used to approximate the normal and other distributions
Lognormal (or gamma)		• System size, effort, complexity • Accounts for typical underestimation of scope and system-wide effects
Truncated lognormal		• To prevent nonpositive values or other inconsistent values from a standard lognormal distribution (e.g., when the mean is close enough to zero that the spread goes negative)

A normal distribution can be used to model some quantities more realistically than a triangular distribution, but it is a symmetric distribution, and care should be taken when using it to ensure that the quantity being modeled is indeed symmetric. It can be used to model errors of various types, other process parameters like size and cost drivers, or quantities that are the sum of a large number of other quantities [according to the Central Limit Theorem (CLT)]. An application of the latter would be the roll-up summary of subhierarchical software components.

The PERT distribution is simple to implement and can approximate a symmetrical normal distribution or a skewed distribution like the triangle or lognormal. It is available in many packages and often gives good enough results given the inherent uncertainties. But if more precise information is known about a parameter distribution, then one should consider an alternative if the PERT is not a good fit and modeling precision is important.

Generally speaking, software engineers are an optimistic bunch when it comes to estimating and want to think that everything will go fine. Also, some aspects of the job are often ignored. Interfaces, glue code, and such are often not considered when estimating a system at the top level, and more often than not the size is underestimated. To compensate for these tendencies, positively skewed distributions of size or effort with long tails are recommended to be realistic for many estimating situations.

The asymmetric log-normal distribution is relevant for modeling system size, complexity, or the time to perform some task, particularly since scope is usually underestimated and unconsidered details tend to increase the job size. It will account for the asymmetry in the expected outcomes. It is often useful for quantities that are the product of a large number of other quantities (by virtue of the CLT).

A.3 MONTE CARLO ANALYSIS

Monte Carlo analysis is a "game of chance" technique used to solve many types of problems by applying random sampling instead of analytic methods. Samples are taken from known input probability distributions to create output distributions. It estimates the likely range of outcomes from a complex random process by simulating the process a large number of times. It can be used to solve stochastic models in simulation, or to obtain approximate solutions for definite integrals, integrodifferential equations, and linear equations. It is an analysis method often referred to as Monte Carlo simulation, which does not necessarily imply continuous system simulation or even time-based simulation (it is often applied to static situations). The term *Monte Carlo analysis* is used to eliminate confusion.

The following steps are performed for n iterations in a Monte Carlo analysis, where an iteration refers to a single simulation run:

1. For each random variable, take a sample from its probability distribution function and calculate its value.
2. Run a simulation using the random input samples and compute the corresponding simulation outputs.
3. Repeat the above steps until n simulation runs have been performed.
4. Determine the output probability distributions of selected dependent variables using the n values from the runs.

A.3.1 Inverse Transform

Monte Carlo analysis uses random numbers to sample from known probability distributions to determine specific outcomes. The inverse transform technique for generating random variates is convenient for this. First, a random number r is chosen that is uniformly distributed between 0 and 1. It is set equal to the cumulative distribution, $F(x) = r$, and x is solved for. A particular value r_0 gives a value x_0, which is a particular sample value of X. It can be expressed as

$$x_0 = F^{-1}(r_0)$$

This construction is shown in Figure A.16 to generate values x of the random variable X that represents size. It graphically demonstrates how the inverse transform uses the cumulative probability function for generating random variates using random samples between 0 and 1 for input.

Referring to Figure A.16, first a random number between 0 and 1 is generated, then the cumulative distribution is used to find the corresponding random variate. In the example from a normal distribution, the first random number of 0.42 generates a value of size = 48. The second random draw of 0.86 produces a value of size = 61 per the figure.

There is a strong intuitive appeal to the inverse transform method. The technique works because more random number inputs will hit the steep parts of the CDF, thus concentrating the random variates under those regions where the CDF is steep, exactly the same regions where the PDF is high. Since the PDF is the derivative of the CDF [$f(x) = F'(x)$], $f(x)$ can be viewed as the slope function of $F(x)$. Thus, the CDF rises most steeply for values of x where $f(x)$ is large and, conversely, it is flat where $f(x)$ is small.

The Monte Carlo method will fill out a distribution of random variates prescribed by the PDF, since most of them will be from the largest areas of the original distribution. The corresponding distribution of random variates will thus resemble the PDF after enough iterations. See the following example for an illustration of Monte Carlo analysis as applied to software effort estimation.

A.3.2 Example: Monte Carlo Analysis

This example will simulate the randomness of the size input to a dynamic effort model, and quantify the resulting output in probabilistic terms. Assume that the likely values for size can be represented with a normal probability distribution as shown in Figure A.17 with a mean of 50 KSLOC and a standard deviation of 10 KSLOC.

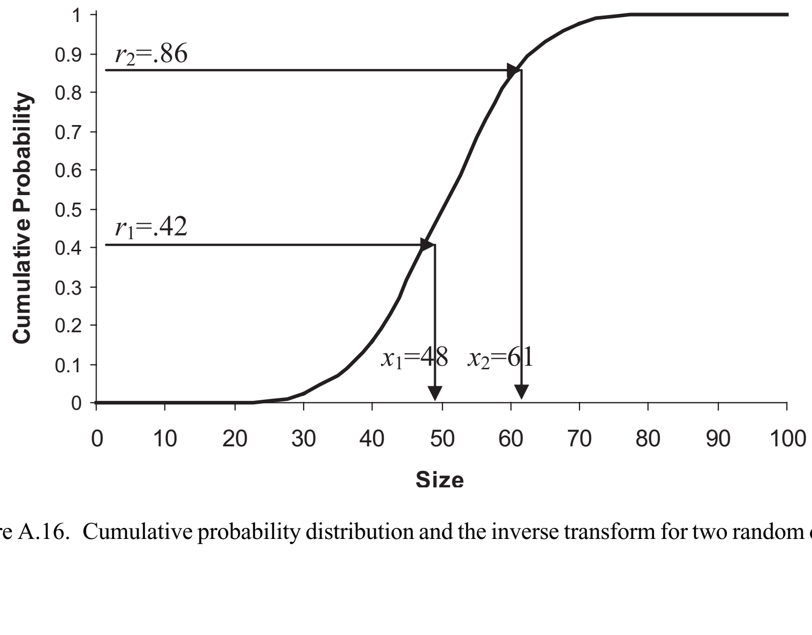


Figure A.16. Cumulative probability distribution and the inverse transform for two random draws.

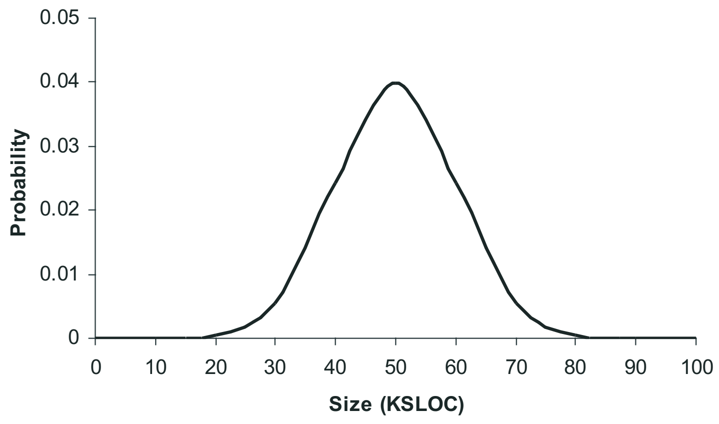


Figure A.17. Normal probability distribution representing size.

To implement Monte Carlo analysis, a small set of $n = 16$ random samples will be generated for size input to the Dynamic COCOMO model. In actual practice, 16 would be a very low number (except for very large and expensive simulation iterations). Generally, 50–100 iterations should be considered as a minimum for filling out distributions, and up to 1000 iterations will give good results in many situations.

First, 16 random numbers between 0 and 1 are generated. The inverse transform technique is then used to determine size by mapping the random numbers onto its cumulative distribution function. Table A.3 shows the set of 16 random numbers (r_i) and the generated size values $F^{-1}(r_i)$.

Table A.3. Monte Carlo analysis inputs

Iteration	Random Number (0–1)	Size (KSLOC)
1	0.321	45.4
2	0.550	51.3
3	0.091	36.6
4	0.807	58.7
5	0.427	48.1
6	0.667	54.3
7	0.451	48.8
8	0.003	22.7
9	0.727	56.0
10	0.360	46.4
11	0.954	66.8
12	0.371	46.7
13	0.224	42.4
14	0.704	55.4
15	0.787	57.9
16	0.202	41.7

Figure A.18 graphically illustrates the inverse transform technique using the same numbers to determine size 16 times via the cumulative distribution function. Note how the steepness in the middle of the CDF produces more size numbers in the central region compared to the more uniform spread of the initial random numbers. The reader is encouraged to eyeball the graph and notice that about 50% of the random spread in the middle maps into about 25% of the values per the distribution.

Dynamic COCOMO is then run 16 times with the respective size inputs. The simulation outputs for the personnel staffing curve in Figure A.34 demonstrate the model sensitivity to the size samples. Project effort is the area under a staffing curve. Table A.4 shows the cumulative project effort for the 16 iterations.

Figure A.20 shows the Monte Carlo results in terms of an effort distribution histogram and its continuous representation. It stacks up the 16 simulation outputs for the total project effort into respective effort bins. A smoother distribution would be seen filled out when more iterations are run and plotted beyond the small sample size of 16.

Figure A.21 shows the cumulative effort distribution used as a confidence level chart. With enough iterations, the cumulative chart could also be drawn on a standard continuous axis without the large bins. One can read that the 80% confidence level is at 290 person-months, which means there is a 20% chance of effort being greater than 290 person-months per the model.

A.3.2.1 Randomness for Multiple Parameters

This simple example only modeled the randomness of a single input parameter (size), and the same technique is used to simulate the random distributions of multiple input parameters at once. Random input samples are drawn for each parameter for each iteration.

Suppose in a given model that the dependent variable effort depends on size and

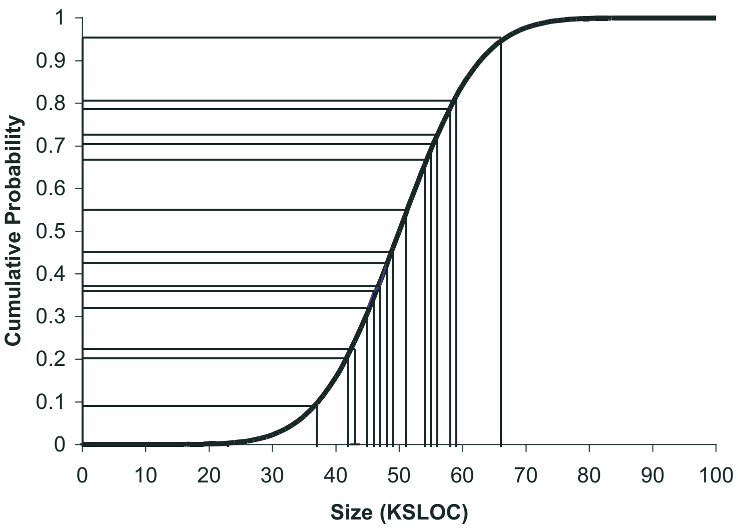


Figure A.18. Monte Carlo analysis using inverse transform technique.

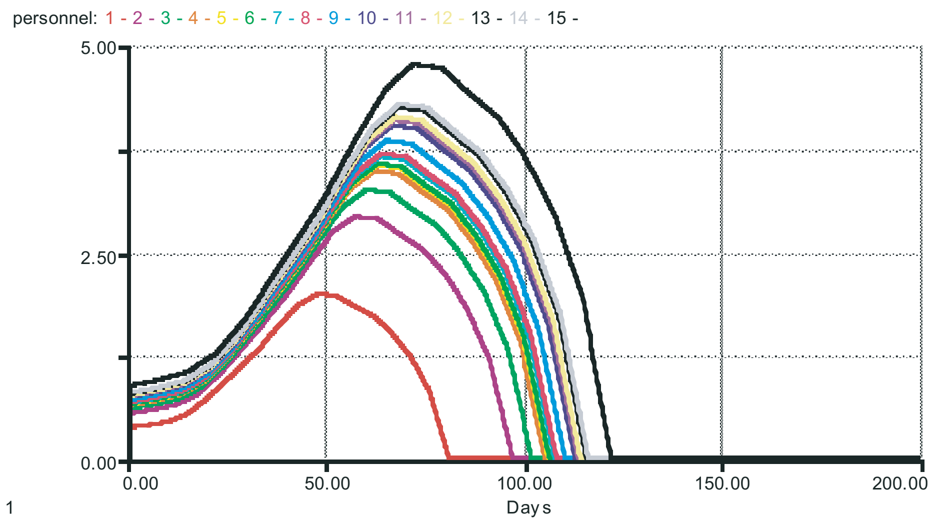


Figure A.19. Monte Carlo results: personnel staffing curves for 16 runs.

complexity:

$$\text{Effort} = f(\text{Size, Complexity})$$

Figure A.22 shows the Monte Carlo procedure for the two-parameter example.

Table A.4. Monte Carlo effort output

Iteration	Effort (Person-months)
1	217.7
2	252.2
3	168.6
4	296.5
5	234.0
6	270.4
7	237.5
8	94.9
9	280.7
10	223.9
11	346.7
12	225.6
13	200.9
14	276.6
15	292.2
16	196.6

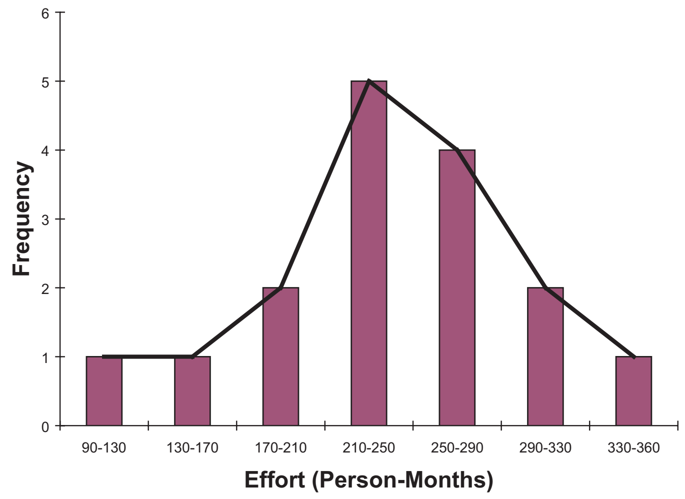


Figure A.20. Monte Carlo results: effort distribution.

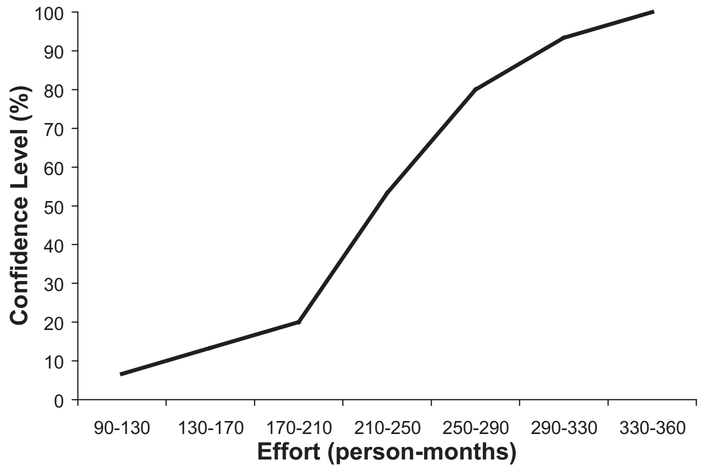


Figure A.21. Monte carlo results: confidence level chart (cumulative effort distribution).

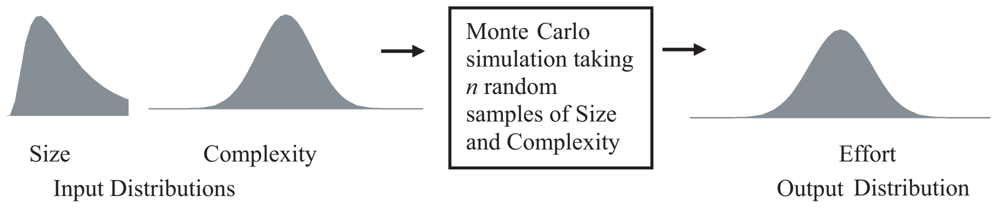

Size
Complexity
Input Distributions
Monte Carlo simulation taking n random samples of Size and Complexity
Effort
Output Distribution

Figure A.22. Monte Carlo analysis for two parameters.

A.4 ANALYSIS OF SIMULATION INPUT

To build a representative simulation model, the random effects in a system must be recreated. This section describes some methods to analyze input data that go hand in hand with the probability concepts previously described. Top-level steps performed during input analysis are:

- Identifying system input variables
- Collecting data on the input variables
- Estimating the parameters of the data distribution
- Fitting known distributions to the data
- Hypothesis testing to validate the distribution
- Selecting a distribution and generating random numbers for the simulation.

One way of recreating random effects is to use a stream of data from real-world processes. There are a number of significant practical problems, though. Field data is usually limited and there is substantial effort and time associated with its collection and reformatting for simulation models. It could also be quite voluminous to use with a model. Field data, if available, is valid only for currently operating processes. However, the purpose of many simulation studies is to analyze hypothetical future systems. Lastly, it could be very difficult to perform sensitivity analyses on field data.

It is important to note that field data could be very useful for validity checking. A model subjected to input field data should ideally perform like the system being simulated, and its output statistics should match closely to the actual system.

Due to the practical limitations described above, a means for artificially generating random data to match certain specifications is desired. You should collect sufficient field data to serve as a reliable sample of the actual statistical population. Next, perform a statistical analysis of the collected sample to identify the probability distribution of the population from which the sample was taken, and then use a method to create random variates that represent the population.

It is desirable to use an available PDF that fits the field data in question. First, one must identify which theoretical PDF best fits the real world data. This can be done by various means such as statistical goodness-of-fit tests. Visual inspections of real data frequency histograms can be used as a first step to identify candidate PDFs before rigorous statistical testing.

A.4.1 Goodness-of-Fit Tests

A goodness-of-fit test is a statistical hypothesis test. It assesses whether data points are independent samples from a particular probability distribution. Two of the more popular goodness-of-fit methods include the older chi-square test and the Kolmogorov–Smirnov test.

In the chi-square test, a histogram of field data is built. The upper and lower bounds of each cell of the histogram are taken, and the probability of the random variable falling within each pair of upper and lower bounds is found. It essentially is a formal comparison of a histogram or line graph with the fitted density function. The following test statistic is calculated:

$$X_2 = \sum_{j=1}^{k} (N_j - np_j)^2/np_j$$

where N_j is the number of X_is in the jth interval, and pj is the expected proportion of of X_is that would fall in the jth interval. Since np_j is the expected number of n X_is that would fall in the jth interval if the hypothesis were true, we would expect X_2 to be small if the fit is good. Therefore, we reject the hypothesis if X_2 is too large.

The Kolmogorov–Smirnov test does not require a histogram of the field data like the chi-square test, and works well with a small number of data points. Its disadvantage is that it only applies to continuous distributions. The test is performed by developing an empirical cumulative probability distribution function based on the field data, and then comparing it with the cumulative probability function of the candidate theoretical distribution. The test is based on the largest absolute deviation between the empirical and theoretical CDF for every given value of x. The following five steps are performed:

1. Rank the field data in increasing order.
2. Compute the following using the theoretical CDF:

$$D^+ = \max_{1 \leqq i \leqq N} [i/N - F(x_i)]$$

$$D^- = \max_{1 \leqq i \leqq N} [F(x_i) - (i-1)/N]$$

3. Let $D = \max(D^+, D^-)$.
4. Find the critical value from a Kolmogorov–Smirnov table for a given significance level and sample size.
5. If D is less than or equal to the critical value, accept the candidate distribution as having a good fit to the field data. Reject it otherwise.

The chi-square test is more appropriate for large numbers of data points and discrete distributions. The Kolmogorov–Smirnov is better suited for smaller sample sizes and continuous distributions. Worked-out examples of these are shown in [Law, Kelton 1991], [Khnoshevis 1994], or other simulation analysis references. Other goodness-of-fit approaches have been developed. The reader is encouraged is study advanced statistical texts to learn more.

A.4.1.1 Example: Goodness-of-Fit Test

We will use the Kolmogorov–Smirnov test to assess whether a set of data on change request sizes is uniformly distributed between 1 and 2 with a significance level of $\alpha = 0.05$. Suppose we have five observations on the size of change requests in function points as follows: 1.38, 1.25, 1.06, 2.0, and 1.56. For the uniform distribution, the CDF is

$$F(x_i) = z/(b - a) \qquad a \leqq x \leqq b$$

When $b = 2$ and $a = 1$, $F(x_i) = z/2$. Table A.5 shows the calculations. $D = \max(0.02, 0.53) = 0.53$. The critical value from Kolmogorov–Smirnov tables for a sample size of 5 and significance level of 0.05 is 0.565. D is less than the critical value, so the hypothesis that the distribution of data is uniform between 1 and 2 is not rejected.

A.5 EXPERIMENTAL DESIGN

An "experiment" in our context is the execution of a computer simulation model. Experimental design is a way of deciding before runs are made which configurations to simulate so that the desired information can be obtained with a minimal amount of simulating. Carefully designed experiments will eliminate the "hit and miss" experience.

Parametric as well as structural changes to the model might be called for. Changing the length of each run, systematically changing the parameters, and multivariate comparisons are involved during experimentation. The experimentation ultimately should lead to an understanding of the simulated system and how to improve it.

In many cases, trial and error is necessary to try a large number of values to optimize system performance. (Note that some system dynamics simulation packages enable limited optimization.) Finding the optimal value for a system may require a heuristic search process using derivatives. There are other special search techniques that are beyond the scope of this book. Heuristic optimization is more valuable as the number of decision parameters increase. Otherwise, all possible combinations of variables must be tried, which can be prohibitively time-consuming.

A simulation model is a mechanism that turns inputs into output performance measures; in this sense, a simulation is just a function. In many cases a three-dimensional response surface can represent the output space of a simulation model. The next section shows an example output response surface. A whole area of response surface methodologies exists to seek the optimal configuration. Many of these techniques use a gradient approach.

Table A.5. Kolmogorov–Smirnov calculations

i	x_i	$F(x_i)$	i/n	$i/n - F(x_i)$	$F(x_i) - (i-1)/n$
1	1.06	0.53	0.2	–0.33	0.53
2	1.25	0.625	0.4	–0.225	0.425
3	1.38	0.69	0.6	–0.09	0.29
4	1.56	0.78	0.8	0.02	0.18
5	2.0	1.0	1.0	0	0.2
				$D^+ = 0.02$	$D^- = 0.53$

The input parameters and structural assumptions are called factors, and the output performance measures are called responses. Which parameters and structures to keep fixed and which are factors to vary depends on the goals of the study rather than the form of a model. Factors are also classified as controllable or uncontrollable depending on whether they represent management options in the real-world system. Normally, we focus on the controllable factors.

The experimental design for a one-factor model is simple. Run the simulation at various values, or levels, of the factor. A confidence interval could be formed for the expected response at each of the factor levels. Suppose now there are k factors and we want an estimate of how each factor affects the response, as well as whether the factors interact with each other. The number of runs can balloon as all factors must be set at specific levels to test their interaction with a particular one. A 2^k factorial design is an economical strategy to measure interactions. It requires only two levels for each factor and then calls for simulation runs at each of the 2^k possible factor-level combinations. The reader should consult advanced statistical texts for more information on experimental design.

A.5.1 Example: Experimental Design and Model Response Surface

The simple Brooks's Law model is Chapter 1 is used to create a model response surface. The model output is defined as schedule gain relative to the original plan of 200 days. The two input parameters varied are the number of added staff and the progress gap threshold used before adding people. The experimental design will simply straddle the region described in the original example around a 15% gap threshold. The model is thus run at the gap thresholds of 5%, 10%, 15%, 20%, and 25%. The added staff varies from 0, 5, and 10 per the original example.

Figure A.23 shows the response surface generated from these experiments. Since schedule gain is plotted in the Z-dimension, a large value is a desired result. Thus,

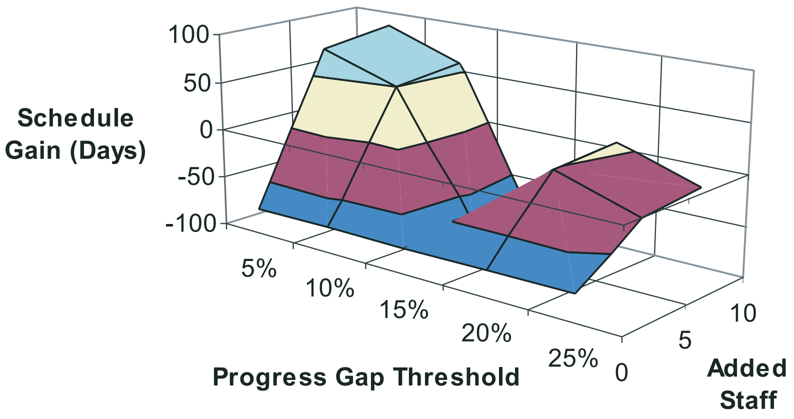


Figure A.23. Brooks's Law model response surface.

maxima can be interpreted as process optima or sweet spots. The behavior of the model can be analyzed from such a response surface.

A.6 ANALYSIS OF SIMULATION OUTPUT

Simulation models convert stochastic inputs and system components into statistical data output. Hence, simulation is another sampling method and the output is subject to statistical analysis. Some purposes of analyzing simulation output are

- To determine characteristics of certain variables for given inputs
- To compare variable characteristics under different conditions
- To improve a system or design a future one

As simulation is a sampling process, variable estimates are subject to sampling choice and sample size. Discrete event simulation has unique concerns that are not addressed here, such as data dependency. Independency of sample data allows use of classical statistical analysis; however, most discrete simulation models lack independence. Queuing processes, for example, usually lead to autocorrelation. Readers who are concerned with data dependency and discrete event modeling are encouraged to read general simulation texts to learn more.

Output analysis typically differs for terminating versus nonterminating systems. In the software process modeling arena, very few applications are for nonterminating systems, hence the distinction will not be addressed here.

A.6.1 Confidence Intervals, Sample Size, and Hypothesis Testing

We desire to know the true value of a simulation output parameter based on the results of many simulation runs. The accuracy of a statistical estimate is expressed over an interval. The confidence level is the probability that the interval contains the true value of the parameter. The confidence interval is updated after the result of each simulation run.

The interpretation of a confidence interval follows. If one constructs a very large number of $(1 - \alpha)$ confidence intervals each based on n observations, in the long run $(1 - \alpha)$ of the intervals contain the parameter value. The parameter α is also called the significance level, or the probability of rejecting a hypothesis when it is true.

For a given confidence level, a small confidence interval is preferable. For a given confidence interval, a higher confidence level is preferable. In practice, choose a confidence level. The sample size affects confidence level and interval, such that smaller confidence intervals require a larger sample size. The $(1 - \alpha)$ confidence interval relationship can be written as

$$P\{-Z_{\alpha/2} \leqq Z \leqq Z_{\alpha/2}\} = 1 - \alpha$$

where Z is distributed normally with a mean of zero and standard deviation of one. Z is expressed as

$$Z = (X - \mu)/\sigma_x$$

where X is the sample mean, μ is the population mean and σ is the standard deviation of the population of random variables. Substituting Z in the previous expression gives the confidence interval as follows:

$$P\{X - Z_{\alpha/2}\sigma/\sqrt{n} \leqq \mu \leqq X + Z_{\alpha/2}\sigma/\sqrt{n}\} = 1 - \alpha$$

Figure A.24 depicts the confidence interval using a normal distribution with a mean of one. The true mean falls in the confidence interval with a probability of $(1 - \alpha)$. Use normal distribution tables for Z if there are greater than 30 samples, and use the student t distribution when the sample size is less than 30.

The formulas for confidence intervals can be manipulated to determine the converse—the required sample size to attain a desired confidence interval width in conjunction with an estimated sample variance. Variance usually decreases with sample size. Other special variance reduction techniques are available, but these are important only for large models and have been deemphasized with modern computing power.

In hypothesis testing, you start with a null hypothesis and set the significance level α (the confidence interval is $1 - \alpha$). Example hypotheses can be that the parameter mean is in an interval or it is greater than a certain value or less than a certain value. You fail to reject the hypothesis if certain statistical relationships hold true. We already saw the formula for interval testing, and the ones for other comparisons are found in statistical texts.

Statistical techniques covered so far assume samples are independent and identically distributed. Independent replication is a technique that addresses the problem of autocorrelation that works as follows:

- Perform several short runs (replications) from time = 0.
- Use different random number seeds per run (or possibly different initial conditions) so replications are then independent of each other.
- Each replication mean is an independent observation.
- Compute the mean, variance, and confidence interval.

Batch means is a method that only applies to steady state analysis. It alleviates the problem of handling transitional periods during independent replications. You divide a single run into multiple intervals (batches). Batch mean values serve as independent samples. You can use the runs up and down test to detect dependencies, and determine optimal batch size to minimize data dependencies. The runs up and down test use sequential data points. A run is a succession of similar increasing or decreasing patterns in the sequence. The number of "runs" is counted, and equations computed for hypothesis testing.

More mathematical background on independent replications, batch means, and runs up and down can be found in [Khoshnevis 1991] or other discrete event simulation

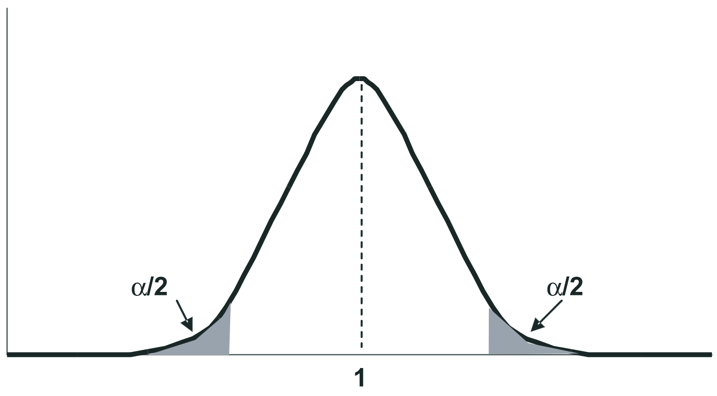


Figure A.24. Confidence interval.

texts. Also beware of nonnormality in the simulation data. This topic becomes quickly advanced and readers should consult more detailed statistics references.

A.6.1.1 Example: Confidence Interval Calculation

We will use the results of the Monte Carlo example simulation in Section A.3.2 to estimate the 90% confidence interval for the effort output. The array of outputs in Table A.4 is used to calculate a mean of 238 and a standard deviation of 58.8. For a sample size of 16, we resort to a standard t-distribution table and find a value of 1.75 corresponding to 15 degrees of freedom and t_{95}. Since we want a 90% interval, $\alpha/2 = 5\%$ and we keep 5% at each tail at the distribution by looking up the value for $1 - \alpha/2$. Thus the 90% confidence interval is

$$P\{238 - 1.75 \cdot 58.8/\sqrt{16}\} \leqq \mu \leqq \{238 - 1.75 \cdot 58.8/\sqrt{16}\} = 0.9 = \{212 \leqq \mu \leqq 264\}$$

Thus the 90% confidence interval for simulated effort is between 212 and 264 person-months, and we fail to reject the hypothesis that the means lies in that interval. The interval would be larger if we wanted 95% confidence.

A.7 MAJOR REFERENCES

[Law, Kelton 1991] Law M., and Kelton W., *Simulation Modeling and Analysis.* New York: McGraw-Hill, 1991.

[Khoshnevis 1992] Khoshnevis B, *Systems Simulation—Implementations in EZSIM.* New York: McGraw-Hill, 1992.

A.8 APPENDIX A SUMMARY

The random effects inherent in systems should be modeled to build representative simulations. It is prudent to look at the range of possible results from a model rather than be blinded by a deterministic point estimate. It is more meaningful to discuss projects in terms of ranges that represent uncertainty, or risk.

In risk analysis, one tries to gauge the adverse impacts and probabilities of risk items. Probability distributions are used to express the stochastic ranges of simulation inputs and outputs and account for the uncertainties. When a probability distribution is used to express the output of several simulation runs, its cumulative form can be used to assess the confidence level for a given value in the distribution. Stakeholders can gauge the associated confidence level of their plans to help risk manage the project and, in fact, every estimate should be reported with an associated confidence level. Otherwise, others should assume the 50% confidence level by default. The overall uncertainty of estimates should decrease over time, however, as a project progresses, because the product becomes more elaborated with less unknowns.

Several popular types of probability distributions can be used for the software process domain, including the uniform, triangle, normal, PERT, and lognormal and its truncated version. Standard parameters are used to describe their shapes. They are best applied in different situations per the guidelines described in the text.

The Monte Carlo method is one of the most convenient and popular ways to create random variates for simulation runs through the use of probability distributions and inverse transforms. Random numbers between 0 and 1 are used to index into a known cumulative probability distribution to generate the random variates. Those variates are then used as inputs to a simulation. Multiple input parameters can be used as random inputs to the Monte Carlo technique to better reflect the uncertainties. There are a variety of tools that can support Monte Carlo simulation.

Simulation input analysis looks at the pattern of real inputs and attempts to model them with appropriate distributions. Goodness-of-fit tests are used to test hypotheses about whether distributions are well fitted to data. The chi-square and Kolmogorov–Smirnov are two frequently used methods for this.

In experimental design, we decide before simulation on what configurations to run. These experiments should be carefully thought out in order to minimize the number of runs and get the maximum information from the simulations. Factors are chosen to be run at different values. A 2^k factorial design is an economical design strategy for larger models.

Simulation is a sampling method in itself and, correspondingly, its output is subjected to statistical analysis. We would like to determine the characteristics of specific outputs with known confidence. Statistical methods are used to determine the confidence intervals for output parameters. Hypothesis testing is used determine if a parameter is an interval or not, after setting the desired significance level and accounting for the sample size.

With proper experimental design, simulation is handy for process optimization to determine "how much is enough." Process models are run over reasonable ranges in order to find the optima. Typically, a balance needs to be found between counteracting effects. Process optimization also helps determine the break-even points where activities lost their cost effectiveness.

Process simulation techniques are crucial for good risk management, and they are highly complementary in nature. Process simulation can help in both the assessment and control of risks. Model factors provide a powerful mechanism to identify risks in union with simulation, so risks are not ignored. Models also support risk management by analyzing cost and schedule impacts, trade-offs, and sensitivities. Simulation can also help quantify the uncertainties in addition to the consequences of risk items. Sensitivity analysis is useful to determine the relative variance of outputs to changes in input parameters. Decision making is also supported by quantifying the risk-reduction leverage of different options.

But analysis is not a one-time event and changes occur. Risk assessment is a continuous process throughout a project life cycle and, commensurately, simulations should be frequently updated and rerun to provide the best information for risk management decisions.

Risk exposure is defined as the probability of a loss multiplied by the cost consequence of that loss. Risk exposure is a convenient framework for software decision analysis, and process modeling can contribute to analyses in several ways. Risk exposure analysis is a valuable framework to find the "sweet spots" of processes, such has how much testing or rigor is enough (Sections 4.6 and 6.4 show examples). By identifying the risks of doing too much and too little, summing risk exposure curves will find the sweet spots.

Process models easily support discrete trade-offs such as choosing between teams or toolsets. The relative costs, schedule, or other impacts can be quantified. Finally, they can help in the back and forth of project negotiations by quantifying the trade space for decisions to be made.

A.9 EXERCISES

A.1. Describe in your own words what a confidence level signifies.

A.2. a) Collect data on important software process measures in your organization and construct their probability distributions. Alternatively, use data from other sources and create representative probability distributions.

b) Now decide if any standard distributions are good fits to the data. Employ goodness-of-fit tests and report your results.

A.3. Here is a dataset of defect fixing times in person-hours: 3.5, 3.0, 17.1, 15.3, 10, 12.9, 4.5, 3.0, 19.4, 21.1, 17.4, 14.5, and 17.1. Construct its frequency histogram and cumulative distribution. Does it appear to be any of the standard distributions?

A.4. Review the basic distributions for modeling software processes. Critically analyze whether they are good representations or not. Suggest other distributions, as applicable.

A.5. Suppose someone tells you an estimate for a software project as a point estimate (single number) with no other information. What is your assumption with no other context given? What questions would you have and what suggestions for improved communication?

A.6. Generate 100 random variates with an exponential distribution and a mean of 15.

A.7. The PDF of function x is defined as

$$f(x) = x/16 \qquad \text{for } 0 \leqq x \leqq 4$$
$$f(x) = 1/4 \qquad \text{for } 4 \leqq x \leqq 6$$

Draw the PDF. Use the inverse transform method to develop the relationships to generate a random variate given a random number between 0 and 1. Describe in your own words how the Monte Carlo analysis technique works.

A.8. Using the inverse transform technique, develop the equations for creating triangularly distributed random variates.

A.9. Use one of the provided models in the book and modify it to run Monte Carlo simulations. Choose an appropriate variable in the model and create your own random variates for it. Make the runs and summarize your results.

A.10. Use a spreadsheet or other tool to create Monte Carlo inputs for (1) a normal, (2) a triangular, and (3) a truncated lognormal distribution.

A.11. Explain the difference in the procedure to determine the 90% confidence interval if 50 observations were available in Section A.6.1.1.

A.12. Construct the 85% and 95% confidence intervals for the Monte Carlo example results.

A.13. Name five specific examples whereby simulation can help in software risk management.

Advanced Exercises

A.14. Apply 2^k experimental design principles to the Chapter 1 Brooks's Law model.

A.15. Modify the Dynamic COCOMO model to accept two random inputs and run a Monte Carlo analysis.