UNIVERSITY OF MINES AND TECHNOLOGY TARKWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTE

CE378 SIMULATION AND MODELING

COMPILED BY

GEORGE ESSAH YAW OKAI

TARKWA, GHANA

MAY, 2021

Objectives Of the Course v
Course Content v
Reference Materials vi
Course Presentation vi
Course Assessment 1
Attendance 1
CHAPTER 1 INTRODUCTION TO MODELING AND SIMULATION 2
1.1 Chapter Objectives and Expected Results 2
1.2 Simulation Modeling 2
1.3 When Simulation is the Appropriate Tool 3
1.4 When Simulation is not an Appropriate Technique 4
1.5 Advantages to Simulation 4
1.6 Disadvantages to Simulation 5
1.7 Areas of Applications of Simulation 6
CHAPTER 2 SYSTEMS AND SYSTEM ENVIRONMENT 8
2.1 Chapter Objectives and Expected Results 8
2.2 Systems and System Environment 8
2.3 System Concept 9
2.3.2 Components of a System 10
2.3.3 Types of Systems 11
2.3.4 Model of a System 13
2.4 Discrete- Event Simulation 15
2.5 Simulation Worldview 16
2.6 Model Building Process 17
2.7 Decisions for Conducting a Simulation Study 19
2.8 Simulation Cost and Risks 22
2.9 EXAMPLE: A Production Control Problem 23
2.10 Project Report 24
2.11 Simple Simulation Example 26
2.12 Overview of Available Software for Simulation Modelling 28
CHAPTER 3 DISCRETE EVENT SIMULATION (DES) 32
3.1 Chapter objectives and expected results 32

3.2 Elements of Discrete Event Simulation 32
3.3 Examples of DES Models 33
3.3.1 Single Machine 34
3.3.2 Single Machine with failures 34
3.3.3 Single Machine with an Inspection Station and Associated Inventory 35
3.4 Monte Carlo Sampling and Histories 36
CHAPTER 4 BASIC MODELING CONCEPTS 38
4.1 Chapter Objectives and Expected Results 38
4.2 Basics of Arena simulation modelling software 38
4.2.1 The Relationship between the Model and Experiment 38
4.2.2 Entities, Attributes, and Process 39
4.2.3 Block diagram and Basic Block Types used in ARENA 39
4.3 The ARENA Window 44
4.3.1 MENU BAR 45
4.3.2 Project Bar 46
4.3.3 Standard Tool Bar 46
4.3.4 Draw and View Bar 47
4.3.5 Animate and Animate Transfer Bar 47
4.3.6 Run Interaction Bar 47
4.3.7 Integration Bar 47
4.3.8 Debug Bar 47
4.4 Building a Model Using Arena 48
4.4.1 EXAMPLE: A simple work station 48
4.4.2 EXAMPLE 2: Simple Model of the Car Wash System 54
4.5 ARENA Data Storage Objects 63
4.5.1 Variables 64
4.5.2 Expressions 64
4.5.3 Attributes 64
4.6 ARENA Output Statistics Collection 65
4.6.1 Statistics Collection via the Statistic Module 65
4.6.2 Statistics Collection via the Record Module 66
4.7 Arena Simulation and Output Reports 67

4.8 EXAMPLE: Two Processes in Series 68
4.9 PRACTICAL ASSIGNMENTS 76
4.9.1 PROBLEM STATEMENT 1 76
4.9.2 PROBLEM STATEMENT 2 76
4.9.3 PROBLEM STATEMENT 3 77

Objectives Of the Course

The objective of this course is to introduce the main concepts of Simulation and Modeling.This course is designed with the specific objectives of helping students:

- To understand basic simulation methods and principles applied to design simulation models.
- To understand a variety of simulation applications for systems
- To design and perform experiments on simulation models
- To analyse simulation output.
- To collect and analyse input data.
- To apply knowledge of Simulation and Modeling to other disciplines

Course Content

A simulation is a computer model that mimics the operation of a real or proposed system. Simulation is a commonly used and a practical technique for modelling and analysing the real operating systems in order to make more effective decisions. Examples of such systems include transportation, supply chain network, job flow, airports, banks, ocean terminals, information systems, emergency response systems. Due to considerable complexity of real systems, many companies find it difficult to investigate the manufacturing and service design and processes without a computer simulation model.

This course is designed to teach students the processes, tools, and techniques for performing effective simulation analyses. In particular, the course focuses on the basic underlying principles of how simulations work, how to collect and analyse input data, how to build basic simulation models using ARENA, how to verify and validate simulation models, and how to interpret and perform statistical analyses of simulation output.

This course is structured as follows:

- Introduction to Simulation and Modeling;
- Basic Modeling Concepts using ARENA modeling software ;
- Model-Based problem Solving
- Model Verification and Validation

- Interpreting Simulation Output
- Discrete Event Simulation (DES)
- Station Sub-models and Entity Transfers
- Modeling Continues Systems etc.

A hands-on lab component will provide students with direct experience on the use of ARENA modeling software commonly used on the market for Simulation and Modelling systems. Classroom lectures will also be augmented with handouts and other materials.

Reference Materials

- Bungartz, H.,J., Zimmer, S., Buchholz, M.and Pflüger, D.(2014) . *Modeling and simulation,* Springer Berlin
- Kelton, W. D., Sadowski, R. P. and Swets, N. B.(2010) ,*Simulation with Arena* , McGraw Hill 5th Edition
- Banks, J., Carson, J., S., Nelson, B., L., and Nicol, D., M.(2005) *Discrete-Event Simulation* Pearson Prentice-Hall 5th Edition
- Altiok,T., and Melamed, B.(2005), "*Simulation Modeling and Analysis with Arena*" 4th Edition

Course Presentation

The course is presented through lectures and Laboratory Hands-on Practice supported with a hand-out. During the Lab sessions, students are guided to solve practical problems with varying degrees of difficulty. Students are also given practical exercise to solve on their own and submit solutions in the form of assignments. The student can best understand and appreciate the subject by attending all lectures and laboratory work, by practising, reading references and hand-outs and by completing all assignments on schedule.

Course Assessment

	Factor	Weight	Location	Date	Time
Grading System	Quizzes (3)	15 %	In class	To Be Announced (TBA)	1 Hour/Each
	Assignments	15%	SD		
	Attendance	10 %	In class	Random	
	Final Exam	60 %	(TBA)	To Be Announced (TBA)	Practical (2hrs)
					Written (2hrs)

80-100% **A**
70-79.9% **B**
60-69.9% **C**
50-59.9% **D**
0-49.9% **FAIL**

Attendance

According to *UMaT* rules and regulations, attendance is mandatory for every student. A random attendance shall be taken to constitute 10% of the total semester mark. The only acceptable excuse for absence is one authorized by the Dean of Students on their prescribed form. However, a student can also ask permission from me to be absent from a particular class with a tangible reason. A student who misses more than 50% of the total attendance marked **WOULD** not be allowed to take the final exams.

CHAPTER 1 INTRODUCTION TO MODELING AND SIMULATION

1.1 Chapter Objectives and Expected Results

The objectives of this chapter is to:

- Define modeling and simulation and discuss the fundamental concepts of modeling and simulation
- Discus the general characteristics of models
- Discus the general characteristics of simulations
- Determine under which circumstances simulations are useful in engineering.

At the end of this chapter, students are expected to have a general understanding of what Modelling and Simulation of a system is; define Simulation and Modelling, know the types of simulation models, simulation applications, and the simulation process.

1.2 Simulation Modeling

Simulation is one of the most powerful analysis tools available to engineers responsible for the design and operation of complex systems. In the ever-increasing competitive world today, simulation has become a very powerful tool for the planning, design, and control of processes and systems.

A ***simulation*** is the imitation of the operation of a ***real-world process*** or system over time. Simulation involves generation of ***artificial history*** of a system and ***drawing inferences***. The behaviour of a system as it evolves over time is studied by developing a ***simulation model***. This model takes the form of a ***set of assumptions*** concerning the operation of the system.

A **computer simulation** is the usage of a computer for the imitation of a real-world process or system. The dynamic responses of one system are represented by the behaviour of another system, which is largely modeled on the former. A simulation requires a model, or a mathematical description of the real system. This is in the form of computer programs, which encompass the key characteristics or behaviours of the selected system. Here, the model is basically a representation of the system and the simulation process is known to depict the operation of the system in time.

Computer simulation replicates real-world events to save researchers time and money in planning for the future.

Modeling *generally refers* to the process of generating a model as a conceptual representation of some phenomenon.

Modeling is the process of producing a model; a model is a representation of the construction and working of some system of interest. A model is similar to but simpler than the system it represents. One purpose of a model is to enable the analyst to predict the effect of changes to the system. On the one hand, a model should be a close approximation to the real system and incorporate most of its salient features. On the other hand, it should not be so complex that it is impossible to understand and experiment with it. A good model is a judicious trade-off between realism and simplicity. Simulation practitioners recommend increasing the complexity of a model iteratively. An important issue in modeling is model validity. Model validation techniques include simulating the model under known input conditions and comparing model output with system output.

Simulation is considered to include both the construction of the model and the experimental use of the model for studying a problem. Therefore, simulation modeling is an experimental and applied methodology that seeks to accomplish the following:

- Describing the behavior of systems,
- Construct theories or hypotheses that account for the observed behavior, and
- Using the model to determine future behavior.

Simulation in general is defined as the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and /or evaluating various strategies for the operation of the system.

What is the difference between simulation and modeling?

Modeling is the act of building a **model**. A **simulation** is the process of using a **model** to study the behavior and performance of an actual or theoretical system. **In a simulation**, **models** can be used to study existing or proposed characteristics of a system.

1.3 When Simulation is the Appropriate Tool

- Simulation enables the study of and experimentation with the internal interactions of a complex system, or of a subsystem within a complex system.

- Informational, organizational and environmental changes can be simulated and the effect of those alternations on the model's behavior can be observed.
- The knowledge gained in designing a simulation model can be of great value toward suggesting improvement in the system under investigation.
- By changing simulation inputs and observing the resulting outputs, valuable insight may be obtained into which variables are most important and how variables interact.
- Simulation can be used as a device to reinforce analytic solution methodologies.
- Simulation can be used to experiment with new designs or policies prior to implementation so as to prepare for what may happen.
- Simulation can be used to verify analytic solutions.
- Simulation models designed for training, allow learning without the cost and disruption of on the-job learning.
- Animation shows a system in simulated operation so that the plan can be visualized.
- The modern systems are so complex that interactions can only be treated only through simulation

1.4 When Simulation is not an Appropriate Technique

- When the problem can be solved by common sense.
- When the problem can be solved analytically.
- If it is easier to perform direct experiments.
- If cost exceed savings
- If resources or time are not available.
- If the system behavior is too complex.
- If no data is available about that system, not even estimates.

1.5 Advantages to Simulation

The following list describes some of the benefits associated with simulation:

1. Developing the model of a system often leads to a better understanding of the real system.
2. Time can be compressed in simulation; years of experience in the real system can be compressed into seconds or minutes.
3. Simulation does not disrupt ongoing activities of the real system.

4. Simulation is far more general than mathematical models and can be used where conditions are not suitable for standard mathematical analysis.
5. Simulation can be used as a game for training experience.
6. Simulation provides a more realistic replication of a system than mathematical analysis.
7. Simulation can be used to analyze transient conditions, whereas mathematical techniques usually cannot.
8. Many standard packaged models, covering a wide range of topics, are available commercially.
9. New policies, operating procedures, decision rules, organizational structures, information flows, etc. can be explored without disrupting ongoing operations.
10. New hardware designs, physical layouts, software programs, transformation systems, etc. can be tested before committing resources to their acquisition and/or implementation.
11. Hypotheses about how or why certain phenomena occur can be tested for feasibility.
12. Insight can be gained about which variables are most important to performance and how these variables interact.
13. Simulation can be used to identify bottlenecks in material, information, and product flow.
14. A simulation study can prove invaluable to understanding how the system really operates as opposed to how everyone thinks it operates.

1.6 Disadvantages to Simulation

With the strength and advantages associated with the use of simulation, there are some disadvantages which are:

1. Although a great deal of time and effort may be spent to develop a model for simulation, there is no guarantee that the model will, in fact, provide good answers.
2. There is no way to prove that a simulation model's performance is completely reliable. Simulation involves numerous repetitions of sequences that are based on randomly generated occurrences. An apparently stable system can, with the right combination of events however unlikely explode.

3. Depending on the system to be simulated, building a simulation model can take anywhere from an hour to 100 worker years. Complicated systems can be very costly and take a long time.
4. Simulation may be less accurate than mathematical analysis because it is randomly based. If a given system can be represented by a mathematical model, it may be better to use than simulation.
5. A significant amount of computer time may be needed to run complex models.
6. The technique of simulation, while making progress, still lacks a standardized approach. Therefore, models of the same system built by different individuals may differ widely.
7. Simulation analysis can be time-consuming and expensive.
8. Simulation results are sometimes difficult to interpret because the model is trying to capture the randomness of the real system; it is hard to determine whether an observation made during a run is due to a significant relationship in the system.
9. Simulation model building requires specialized training. The quality of the analysis depends on the quality of the model and the skill of the modeler.

1.7 Areas of Applications of Simulation

Some examples of application areas of simulation are as follows:

1. Estimating a set of productivity measures in production systems, inventory systems, manufacturing processes, materials handling, and logistics operations
2. Designing and planning the capacity of computer systems and communication networks so as to minimize response times
3. Conducting war games to train military personnel or to evaluate the efficacy of proposed military operations
4. Evaluating and improving maritime port operations, such as container ports or bulk material marine terminals (coal, oil, or minerals), aimed at finding ways of reducing vessel port times
5. Improving health care operations, financial and banking operations, and transportation systems and airports, among many others.

6. Dynamic modeling of continuous manufacturing systems, using analogies to electrical systems.
7. Benchmarking of a stochastic production planning model in simulation test bed.
8. Paint line color change reduction in automobile assembly plant.

CHAPTER 2 SYSTEMS AND SYSTEM ENVIRONMENT

2.1 Chapter Objectives and Expected Results

The objective of this chapter is to:

- Establish the fundamental Concepts of Systems and System Environment
- Identify general characteristics of a Model of a System
- To understand Model Building Process

At the end of this chapter, students are expected to have a general understanding of the fundamental Concepts of Systems and System Environment; identify the types of systems Simulation and Modeling techniques can be used in , know the types of simulation models, simulation applications, and the simulation processes.

2.2 Systems and System Environment

Before we are to model anything, we need to understand some basic concepts. The two basic concepts are:

- System; and
- System environment

A system is defined as a group of objects that are joined together in some regular interaction toward the accomplishment of some purpose.

A system is a group of interacting or interrelated elements that act according to a set of rules to form a unified whole. A system, surrounded and influenced by its environment, is described by its boundaries, structure and purpose and expressed in its functioning. A system has various inputs, which go through certain processes to produce certain outputs, which together, accomplish the overall desired goal for the system

An example of a system is the automobile factory where the group of objects are: Machines, Component parts and workers operate jointly along assembly line.

A system is often affected by changes that are occurring outside the system this is what is called **system environment.**

In order to understand the relationship between inputs, outputs and processes, you need to understand the environment in which all of this occurs.

The environment represents everything that is important to understanding the functioning of the system, but is not part of the system. The environment is that part of the world that can be ignored in the analysis except for its interaction with the system.

In a factory system, arrival orders can be considered as outside of the influence of the factory. Therefore, it can be part of the environment. If the effects of supply on demand are important, then this must be part of the system.

2.3 System Concept

There are various concepts related to systems under examination which as follows:

System: An organized entity made up of interrelated and interdependent parts.

Boundaries: these are barriers that define a system and distinguish it from other systems in the environment.

Homeostasis: the tendency of a system to be resilient towards external factors and maintain its key characteristics.

Adaptation: the tendency of a self-adapting system to make the internal changes needed to protect itself and keep fulfilling its purpose.

Reciprocal transactions: circular interactions that systems engage such that they influence one another.

Feedback Loop: the process by which systems self-correct based on reactions from other systems in the environment.

Throughput: rate of energy transfer between the system and its environment during the time it is functioning.

Microsystem: the system closest to the client.

Mesosystem: relationships among the systems in an environment.

Ecosystem: a relationship between two systems that has an indirect effect on a third system.

Macrosystem: a larger system that influences clients, such as policies, administration of entitlement programs and culture.

2.3.2 Components of a System

Once we know about system and system environment, there are some other concepts that define the components of a system:

- **Entity**: an entity is an object of interest in a system.

  Examples: in the factory system, departments, orders, parts, and products are the entities.

- **Attributes:** An attribute denotes the property of an entity.

  Examples: Quantities for each order, type of parts or number of machines in a department.

- **Activity:** Any process causing changes in a system.

  Example: Manufacturing process of a department

- **State of the system:** the state of the system is defined as the collection of variables necessary to describe a system at any time, relative to the objective of study. In order words, state of the system means the description of all the entities, attributes and activities as they exist at one point in time,

- **Event:** An event is defined as an instantaneous occurrence that may change the state of the system.

- **Endogenous System**: the term endogenous is used to describe activities and events occurring within a system.

  Example drawing cash in a bank.

- **Exogenous System:** The term exogenous is used to describe activities and events in the environment that affect the system.

  Example arrival of customers

- **Closed System:** A system for which there is no exogenous activity and event is said to be a closed system.

  Example water in an insulated flask

- **Open System:** A system for which there is exogenous activity and event is said to be an open system.

Example of a system and its components:

- Banking system

  Entities: customers

  Attributes: customer arrival time, account balance

  Activity: making deposits, withdrawal

  Events: arrival and departure

  State variables: number of customers waiting (queue)

2.3.3 Types of Systems

In the real world, things are complex. Therefore, measuring the exact values is important for us to develop the most appropriate model. Computers by themselves are based on a digital circuit and as such, intrinsically, their internal representation is based on discrete steps. There are broadly two implementations of systems in the real world:

- *Discrete System; and*
- *Continuous System*

Few systems in practice are wholly discrete or continuous, but since one type of change predominates for most systems, it will usually be possible to classify (Low and Kelton, 2009).

Discrete System: A discrete system is one in which the state variables change only at discrete set of points in time.

Examples:

- Customer arrival in a bank

The state variable number of customers changes only when a customer arrives or when the service provided a customer is completed as shown in figure 1.1.

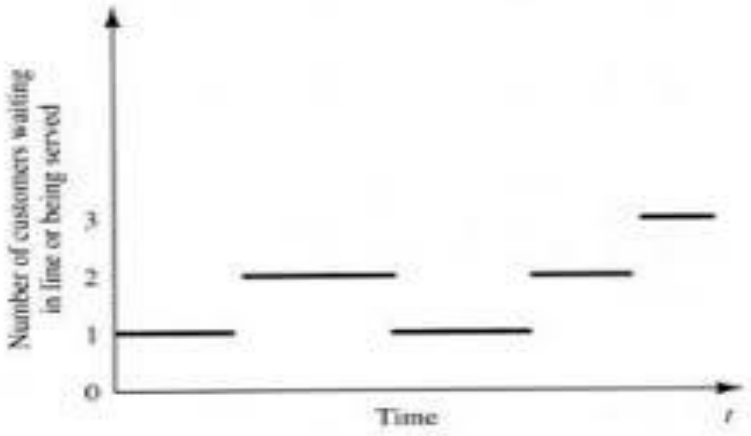


Figure 1.1. Discrete-system state variable.

Continuous System: Continuous system is the one in which the state variables change continuously over time.

Example:

- Head of Water behind a dam

The state variable head of Water changes as rain comes in or dam is emptied etc as shown in figure 1.2.

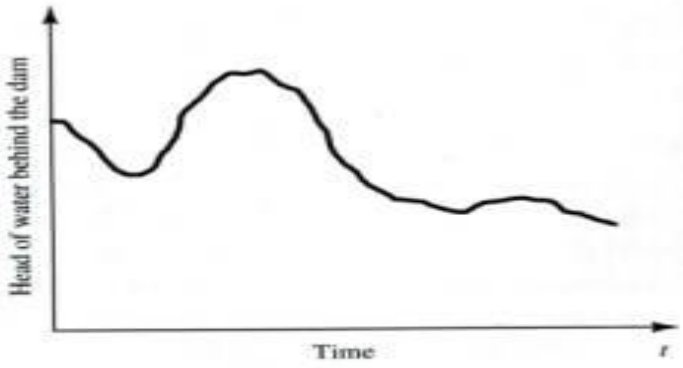


Figure 1.2. Continuous-system state variable.

2.3.4 Model of a System

Modeling is the enterprise of devising a simplified representation of a complex system with the goal of providing predictions of the system's performance measures (metrics) of interest. Such a simplified representation is called a ***model***.

A model is the representation of a real-world system for the purpose of studying the system.

A model is designed to capture certain behavioral aspects of the modeled system those that are of interest to the analyst/modeler in order to gain knowledge and insight into the system's behavior (Morris 1967).

Models are used because we want to learn something about some real system that we cannot observe or experiment with directly either because the system does not yet exist, or it is too difficult to manipulate.

Modeling calls for abstraction and simplification and therefore models are typically built to avoid the situation where the model cost may approach that of the modeled system, thereby militating against creating a model in the first place. More specifically, while modeling is ultimately motivated by economic considerations, several motivational strands may be discerned:

- ***Evaluating system performance under ordinary and unusual scenarios***: A model may be a necessity if the routine operation of the real-life system under study cannot be disrupted without severe consequences e.g., attempting an upgrade of a production line in the midst of filling customer orders with tight deadlines.
- ***Predicting the performance of experimental system designs***: When the underlying system does not yet exist, model construction and manipulation is far cheaper and safer than building the real-life system or even its prototype.
- ***Ranking multiple designs and analyzing their tradeoffs:*** This case is related to the previous one, except that the economic motivation is even greater. It often arises when the requisition of an expensive system with detailed specifications is awarded to the bidder with the best cost–benefit metrics.

The first step in creating any model is to specify its purpose. The model developed should be as simple as the stated purpose will allow.

To be effective, simulation modelling must concentrate on some previously defined problem, otherwise elements to include in the model or what information to generate and collect is not known.

Sample models are used for the following:

1. *To predict and compare, i.e., to provide a logical way of forecasting the outcomes that follow alternative actions or decisions and to indicate a preference among them.*
2. *Model building also provides a systematic, explicit, and efficient way to focus judgment and intuition and*
3. *Also, simulation models can effectively communicate system configuration and assist the thought process.*

Models can assume a variety of forms:

From the top down level models are of two types:

- ***A physical model*** *is a simplified or scaled-down physical object (e.g., scale model of an airplane).*
- ***A mathematical or analytical model*** *is a set of equations or relations among mathematical variables (e.g., a set of equations describing the workflow on a factory floor). It uses symbolic notation and mathematical equations to represent a system.*

What is a simulation model?

A simulation model is a special kind of mathematical or analytical model which can be classified as being static or dynamic, deterministic or stochastic, and discrete or continuous.

Static simulation model: *Also known as a Monte Carlo Simulation, represents a system at a particular point of time.*

Dynamic simulation model*: represents systems as they change overtime. Example is simulation of a bank from 6 am to 5 pm.*

Deterministic model*: Simulation models that contain no random variables are classified as deterministic. Deterministic models have known set of inputs, which will result in known set of*

outputs. Example is deterministic arrivals would occur at a dentist's office if all patients arrive at the scheduled appointed time.

Stochastic simulation model: *Has one or more random variables as inputs. These random inputs lead to random outputs. Random outputs can be considered only as estimates of the true characteristics of a model. An example is the simulation of a bank would usually involve random interarrival time and random service times.*

Discrete and Continuous simulation models: *are defined in an analogous manner. However, discrete simulation model is not used to model a discrete system, nor a continuous simulation model used to model a continuous system. Example tanks and pipes are modeled discretely by some software vendors, even though we know that fluid flow is continuous.*

In addition, simulation models may be mixed, both discrete and continuous. The choice of whether to use both discrete and continuous simulation models is a function of:

- *Characteristics of the system*
- *The objective of the study*

2.4 Discrete- Event Simulation

Discrete-Event systems simulation is the modelling of systems in which the state variable only changes at a discrete set of points in time. The simulation models are analysed by numerical methods rather than by analytical methods. Analytical methods employ deductive reasoning of mathematics to solve the model.

Example: differential calculus can be used to compute the minimum-cost policy for some inventory model.

Numerical methods employ computational procedures to solve mathematical models. In the case of simulation models, which employ numerical methods, models are run rather than solved.

In using numerical methods:

- An artificial history of the system is generated from the model assumptions.
- Observations are collected to be analyzed and to estimate the true system performance measures.

Real world simulation models are rather large. The amount of data stored and manipulated is vast, so such runs are usually conducted with the aid of a computer. However, much insight can be obtained by simulating small models manually.

2.5 Simulation Worldview

A worldview is a set of ideas relating to how a particular computer tool was developed. Every computer tool has two associated worldviews:

- *A developer worldview and*
- *A user worldview.*

The first worldview pertains to the philosophy adopted by the creators of the simulation software tool (in our case, software designers and engineers). The second worldview pertains to the way the system is employed as a tool by end-users (in our case, analysts who create simulation models as code written in some simulation language). A system worldview may or may not coincide with an end-user worldview, but the latter includes the former.

The majority of modern computer simulation tools implement a system worldview, called the ***discrete-event simulation*** (DES) paradigm. In this system worldview, the simulation model possesses a state at any point in time. The state trajectory over time is abstracted as a piecewise-constant function, whose jumps (discontinuities) are triggered by discrete events. More simply, the simulation state remains unchanged unless a simulation event occurs, at which point the model undergoes a state transition. The model evolution is governed by a clock and a chronologically ordered event list. Each event is implemented as a procedure (computer code) whose execution can change state variables and possibly schedule other events. A simulation run is started by placing an initial event in the event list, proceeds as an infinite loop that executes the current most imminent event (the one at the head of the event list), and ends when an event stops or the event list becomes empty. This beguilingly simple paradigm is extremely general and astonishingly versatile. Early simulation languages employed a user worldview that coincided with the discrete-event paradigm. A more convenient, but more specialized, paradigm is the transaction-driven paradigm (commonly referred to as process orientation). In this popular paradigm, there are two kinds of entities: transactions and resources. A resource is a service-providing entity, typically stationary in space (e.g., a machine on the factory floor). A transaction is a mobile entity that moves among “geographical” locations (nodes). A transaction may experience delays while

waiting for a resource due to contention (e.g., a product that moves among machines in the course of assembly). Transactions typically go through a life cycle: they get created, spend time at various locations, contend for resources, and eventually depart from the system. The computer code describing a transaction's life cycle is called a process. Queuing elements figure prominently in this paradigm, since facilities typically contain resources and queues. Accordingly, performance measures of interest include statistics of delays in queues, the number of transactions in queues, utilization, uptimes and downtimes of resources subject to failure, and lost demand, among many others.

2.6 Model Building Process

The purpose of simulation modelling is to help the decision maker solve a particular problem. Therefore, to be a good simulation modeller, one must merge good problem-solving techniques with good software-engineering practice.

The following steps should be taken in every simulation study:

1. ***Problem analysis and information collection***: The first step in building a simulation model is to analyze the problem itself. Note that system modeling is rarely undertaken for its own sake. Rather, modeling is prompted by some system-oriented problem whose solution is the mission of the underlying project. In order to facilitate a solution, the analyst first gathers structural information that bears on the problem, and represents it conveniently. This activity includes the identification of input parameters, performance measures of interest, relationships among parameters and variables, rules governing the operation of system components, and so on. The information is then represented as logic flow diagrams, hierarchy trees, narrative, or any other convenient means of representation. Once sufficient information on the underlying system is gathered, the problem can be analyzed and a solution mapped out.
2. ***Data collection***: Data collection is needed for estimating model input parameters. The analyst can formulate assumptions on the distributions of random variables in the model. When data are lacking, it may still be possible to designate parameter ranges, and simulate the model for all or some input parameters in those ranges.
3. ***Model construction***: Once the problem is fully studied and the requisite data collected, the analyst can proceed to construct a model and implement it as a computer program. The

computer language employed may be a general-purpose language (e.g., C++, Visual Basic, and FORTRAN) or a special-purpose simulation language or environment (e.g., Arena, Promodel, GPSS).

4. ***Model verification***: The purpose of model verification is to make sure that the model is correctly constructed. Differently stated, verification makes sure that the model conforms to its specification and does what it is supposed to do. Model verification is conducted largely by inspection, and consists of comparing model code to model specification. Any discrepancies found are reconciled by modifying either the code or the specification.
5. ***Model validation***: Every model should be initially viewed as a mere proposal, subject to validation. Model validation examines the fit of the model to empirical data (measurements of the real-life system to be modeled). A good model fit means here that a set of important performance measures, predicted by the model, match or agree reasonably with their observed counterparts in the real-life system. Of course, this kind of validation is only possible if the real-life system or emulation thereof exists, and if the requisite measurements can actually be acquired. Any significant discrepancies would suggest that the proposed model is inadequate for project purposes, and that modifications are called for. In practice, it is common to go through multiple cycles of model construction, verification, validation, and modification.
6. ***Designing and conducting simulation experiments***: Once the analyst judges a model to be valid, he or she may proceed to design a set of simulation experiments (runs) to estimate model performance and aid in solving the project's problem (often the problem is making system design decisions). The analyst selects a number of scenarios and runs the simulation to glean insights into its workings. To attain sufficient statistical reliability of scenario-related performance measures, each scenario is replicated (run multiple times, subject to different sequences of random numbers), and the results averaged to reduce statistical variability.
7. ***Output analysis***: The estimated performance measures are subjected to a thorough logical and statistical analysis. A typical problem is one of identifying the best design among a number of competing alternatives. A statistical analysis would run statistical inference tests to determine whether one of the alternative designs enjoys superior performance measures, and so should be selected as the apparent best design.

8. ***Final recommendations***: Finally, the analyst uses the output analysis to formulate the final recommendations for the underlying systems problem. This is usually part of a written report.

2.7 Decisions for Conducting a Simulation Study

Completing the required steps of a simulation study establishes the likelihood of the study's success. Although knowing the basic steps in the simulation study is important, it is equally important to realize that not every problem should be solved using simulation. In the past, simulation required the specialized training of programmers and analysts dedicated to very large and complex projects. Now, due to the large number of software available, simulation at times is used inappropriately by individuals lacking the sufficient training and experience. When simulation is applied inappropriately, the study will not produce meaningful results. The failure to achieve the desired goals of the simulation study may induce blaming the simulation approach itself when in fact the cause of the failure lies in the inappropriate application of simulation.

To recognize if simulation is the correct approach to solving a particular problem, four items should be evaluated before deciding to conduct the study:

- *Type of Problem*
- *Availability of Resources*
- *Costs*
- *Availability of Data*

1. *Type of Problem*: If a problem can be solved by common sense or analytically, the use of simulation is unnecessary. Additionally, using algorithms and mathematical equations may be faster and less expensive than simulating. Also, if the problem can be solved by performing direct experiments on the system to be evaluated, then conducting direct experiments may be more desirable than simulating. To illustrate, recently the UH Transportation Department conducted field studies on expanding the campus shuttle system. The department used their own personnel and vehicles to perform the experiment during the weekend. In contrast, developing the simulation model for the shuttle system took one student several weeks to complete. However, one factor to consider when performing direct experiments is the degree in which the real system will be disturbed. If

a high degree of disruption to the real system will occur, then another approach may be necessary. The real system itself plays another factor in deciding to simulate. If the system is too complex, cannot be defined, and not understandable then simulation will not produce meaningful results. This situation often occurs when human behavior is involved.

2. *Availability of Resources*: People and time are the determining resources for conducting a simulation study. An experienced analyst is the most important resource since such a person has the ability and experience to determine both the model's appropriate level of detail and how to verify and validate the model. Without a trained simulator, the wrong model may be developed which produces unreliable results. Additionally, the allocation of time should not be so limited so as to force the simulator to take shortcuts in designing the model. The schedule should allow enough time for the implementation of any necessary changes and for verification and validation to take place if the results are to be meaningful.
3. *Costs*: Cost considerations should be given for each step in the simulation process, purchasing simulation software if not already available, and computer resources. Obviously if these costs exceed the potential savings in altering the current system, then simulation should not be pursued.
4. *Availability of Data*: The necessary data should be identified and located, and if the data does not exist, then the data should be collectible. If the data does not exist and cannot be collected, then continuing with the simulation study will eventually yield unreliable and useless results. The simulation output cannot be compared to the real system's performance, which is vital for verifying and validating the model.

The basic steps and decisions for a simulation study are incorporated into a flowchart as shown below:

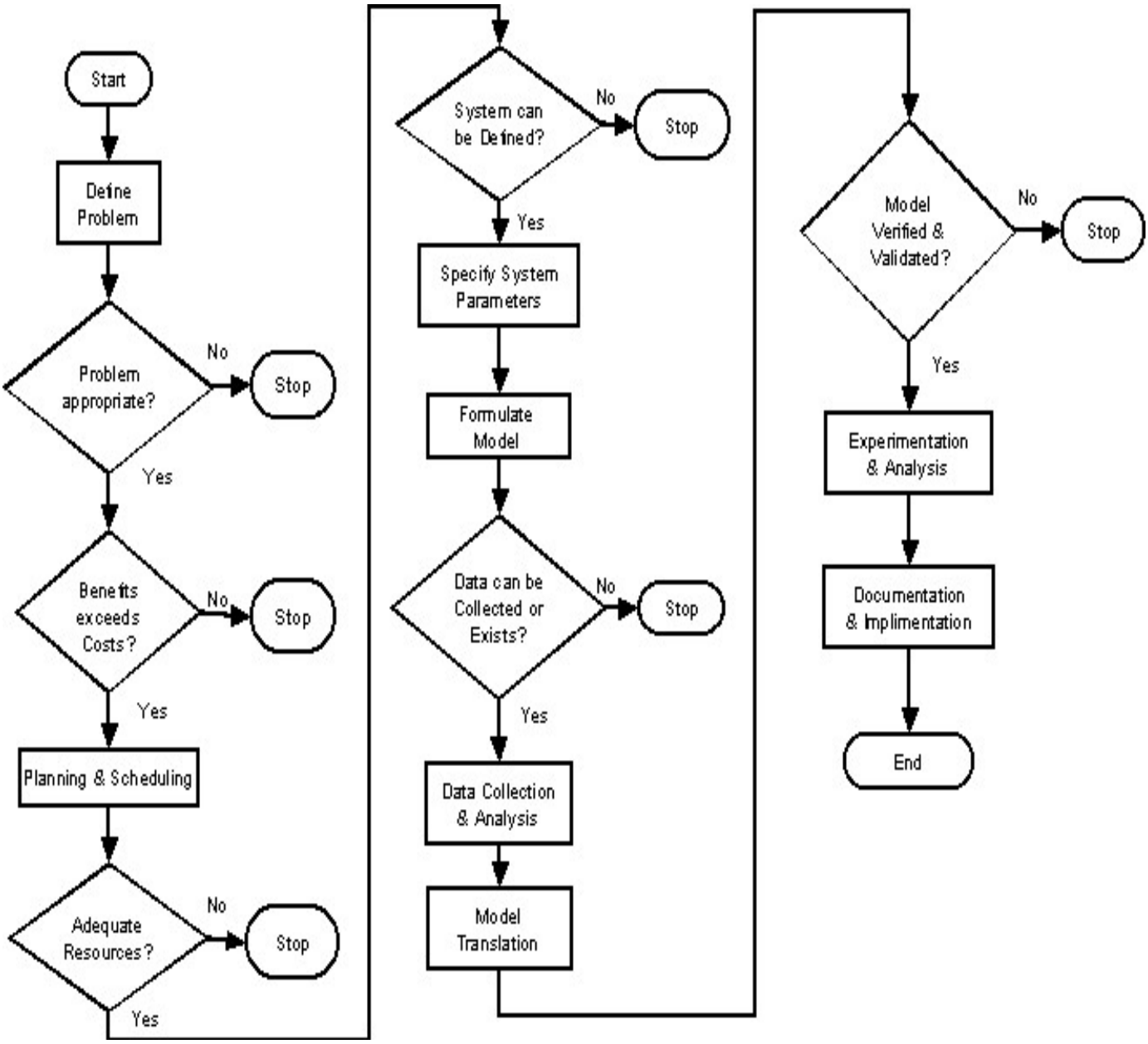


Figure 1.3 Steps and Decisions for Conducting a Simulation Study

Once simulation has been identified as the preferred approach to solving a particular problem, the decision to implement the course of action suggested by the simulation study's results does not necessarily signify the end of the study, as indicated in the flowchart above. The model may be maintained to check the system's response to variability experienced by the real system. However, the extent to which the model may be maintained largely depends on the model's flexibility and what questions the model was originally designed to address.

2.8 Simulation Cost and Risks

Simulation modeling, while generally highly effective, is not free. The main costs incurred in simulation modeling, and the risks attendant to it, are listed below.

- ***Modeling cost***: Like any other modeling paradigm, good simulation modeling is a prerequisite to efficacious solutions. However, modeling is frequently more art than science, and the acquisition of good modeling skills requires a great deal of practice and experience. Consequently, simulation modeling can be a lengthy and costly process. This cost element is, however, a facet of any type of modeling. As in any modeling enterprise, the analyst runs the risk of postulating an inaccurate or patently wrong model, whose invalidity failed to manifest itself at the validation stage.
- ***Coding cost***: Simulation modeling requires writing software. This activity can be error prone and costly in terms of time and human labor (complex software projects are notorious for frequently failing to complete on time and within budget). In addition, the ever-present danger of incorrect coding calls for meticulous and costly verification.
- ***Simulation runs***: Simulation modeling makes extensive use of statistics. The analyst should be careful to design the simulation experiments, so as to achieve adequate statistical reliability. This means that both the number of simulation runs (replications) and their length should be of adequate magnitude. Failing to do so is to risk the statistical reliability of the estimated performance measures. On the other hand, some simulation models may require enormous computing resources (memory space and CPU time). The modeler should be careful not to come up with a simulation model that requires prohibitive computing resources.

- ***Output analysis***: Simulation output must be analyzed and properly interpreted. Incorrect predictions, based on faulty statistical analysis, and improper understanding of system behavior are ever-present risks.

2.9 EXAMPLE: A Production Control Problem

A simple production control problem is represented as an example of the kind of systems amenable to simulation modeling. This example illustrates system definition and associated performance issues.

Consider a packaging/warehousing process with the following steps:

1. The product is filled and sealed.
2. Sealed units are placed into boxes and stickers are placed on the boxes.
3. Boxes are transported to the warehouse to fulfill customer demand.

These steps can be combined into a single processing time, as depicted in the system schematic of Figure 1.4.

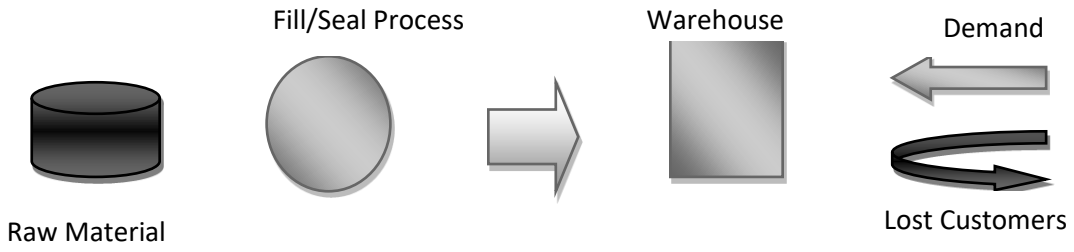


Figure 1.4 Pictorial representation of a packaging/warehousing system

The system depicted in Figure 1.2 is subject to the following assumptions:

1. There is always sufficient raw material for the process never to starve.
2. Processing is carried out in batches, five units to a batch. Finished units are placed in the warehouse. Data collected indicate that unit-processing times are uniformly distributed between 10 and 20 minutes.
3. The process experiences random failures, which may occur at any point in time. Times between failures are exponentially distributed with a mean of 200 minutes. Data collection also showed that repair times are normally distributed, with a mean of 90 minutes and a standard deviation of 45 minutes.
4. The warehouse has a capacity (target level) of R ¼ 500 units. Processing stops when the inventory in the warehouse reaches the target level. From this point on, the production process becomes blocked and remains inactive until the inventory level drops to the reorder

point, which is assumed to be r ¼ 150 units. The process restarts with a new batch as soon as the reorder level is down-crossed. This is a convenient policy when a resource needs to be distributed among various types of products. For instance, when our process becomes blocked, it may actually be assigned to another task or product that is not part of our model.

5. Data collection shows that inter-arrival times between successive customers are uniformly distributed between 3 and 7 hours, and that individual demand sizes are distributed uniformly between 50 and 100 units. On customer arrival, the inventory is immediately checked. If there is sufficient stock on hand, that demand is promptly satisfied; otherwise, customer demand is either partially satisfied and the rest is lost (that is, the unsatisfied portion represents lost business), or the entire demand is lost, depending on the availability of finished units and the loss policy employed.

This problem is also known as a production/inventory problem. We mention, however, that the present example is a gross simplification. More realistic production/ inventory problems have additional wrinkles, including multiple types of products, production setups or startups, and so on. Some design and performance issues and associated performance measures of interest follow:

1. Can we improve the customer service level (percentage of customers whose demand is completely satisfied)?
2. Is the machinery underutilized or over utilized (machine utilization)?
3. Is the maintenance level adequate (downtime probabilities)?
4. What is the tradeoff between inventory level and customer service level?

2.10 Project Report

Once a system has been modeled, simulated, and analyzed, the study and its conclusions are often written up as a project report. A well-thought-out and properly written report is essential to the success of the study, since its value lies in its conclusions and their dissemination to technical and management personnel. A project report should be clearly and plainly written, so that nontechnical people as well as professionals can understand it. This is particularly true for project conclusions and recommendations, as these are likely to be read by management as part of the decision-making process.

A generic simulation project report addresses the model building stages described and consists of the following sections:

- ***Cover page***: Includes a project title, author names, date, and contact information in this order.
- ***Executive summary***: Provides a summary of the problem studied, model findings, and conclusions.
- ***Table of contents***: Lists section headings, figures, and tables with the corresponding page numbers.
- ***Introduction***: Sets up the scene with background information on the system under study (if any), the objectives of the project, and the problems to be solved. If appropriate, include a brief review of the relevant company, its location, and products and services.
- ***System description***: Describes in detail the system to be studied, using prose, charts, and tables.
- ***Input analysis***: Describes empirical data collection and statistical data fitting. For example, the Arena Input Analyzer provides facilities for fitting distributions to empirical data and statistical tests.
- ***Simulation model description***: Describes the modeling approach of the simulation model, and outlines its structure in terms of its main components, objects, and the operational logic. Be sure to decompose the description of a complex model into manageable-size sub-model descriptions. Critical parts of the model should be described in some detail.
- ***Verification and validation***: Provide supportive evidence for model goodness via model verification and validation to justify its use in predicting the performance measures of the system under study.

  To this end, be sure to address at least the following two issues:

  (1) Does the model appear to run correctly and to provide the relevant statistics (verification)?

  (2) If the modeled system exists, how close are its statistics (e.g., mean waiting times, utilizations, throughputs, etc.) to the corresponding model estimates (validation)?
- ***Output analysis***: Describes simulation model outputs, including run scenarios, number of replications, and the statistical analysis of simulation-produced observations. For example, the Arena Output Analyzer provides facilities for data graphing, statistical estimation, and statistical comparisons, while the Arena Process Analyzer facilitates parametric analysis of simulation models.

- ***Simulation results***: Collects and displays summary statistics of multiple replicated scenarios. Arena's report facilities provide the requisite summary statistics.
- ***Suggested system modifications (if any):*** A common motivation for modeling an extant system is to come up with modifications in system parameters or configurations that produce improvements, such as better performance and reduced costs. Be sure to discuss thoroughly the impact of suggested modifications by quantifying improvements and analyzing tradeoffs where relevant.
- ***Conclusions and recommendations***: Summarize study findings and furnishes a set of recommendations.
- ***Appendices***: Contains any relevant material that might provide undesired digression in the report body.

Needless to say, the modeler/analyst should not hesitate to customize and modify the previous outlined skeleton as necessary.

2.11 Simple Simulation Example

A very simple example demonstrating the concept of simulation is that supposing there is a single channel queuing (waiting line) system, such as a checkout customer in a drug store. Assume that the time between counter arrivals is uniformly distributed from 1 to 10 minutes. Let us assume further, that the time to service a customer is also uniformly distributed between 1 and 6 minutes. A simulation model can be developed to determine the average total time a customer spends being checked out (both waiting line plus service time) and the percentage of time that the clerk is idle.

SOLUTION:

To simulate this system given above, there must be a way of generating artificial experience that is characteristic of the situation. Therefore, there must be the need to generate artificial arrival times for each customer and service times for each customer; also there must be a good bookkeeping system to keep track of what happens. A spinner dial with the face divided into 10 equal and numbered parts is used to generate the time between arrivals of customers and a single die is used to generate the service times.

The simulation clock is started at time 00:00and then the dial is spinned to see how many minutes until the first customer comes up to the counter. The die is then rolled to find out how long it will

take to service this customer. By continuing this process for subsequent customers and setting up a bookkeeping system to keep track of when a customer arrives, begins service, and ends service as well as when the system is idle, an information resembling that in the following table below for the first 20 customers.

Customer	Time Between Arrivals	Service Time	Arrival Clock Time	Service		Time in System	Idle Time
				Begins	Ends		
1	-	3	00:00	00:00	00:03	3	0
2	6	1	00:06	00:06	00:07	1	3
3	6	6	00:12	00:12	00:18	6	5
4	4	4	00:16	00:18	00:22	6	0
5	7	2	00:23	00:23	00:25	2	1
6	1	4	00:24	00:25	00:29	5	0
7	1	5	00:25	00:29	00:34	9	0
8	9	4	00:34	00:34	00:38	4	0
9	8	3	00:42	00:42	00:45	3	4
10	9	1	00:51	00:51	00:52	1	6
11	6	6	00:57	00:57	01:03	6	5
12	5	4	01:02	01:03	01:07	5	0
13	4	5	01:06	01:07	01:12	6	0
14	8	1	01:14	01:14	01:15	1	2
15	8	4	01:22	01:22	01:26	4	7
16	7	2	01:29	01:29	01:31	2	3
17	4	2	01:33	01:33	01:35	2	2
18	2	6	01:35	01:35	01:41	6	0
19	7	1	01:42	01:42	01:43	1	1
20	9	4	01:51	01:51	01:55	4	8
				TOTAL		77	47

Average Time in System = 77/20 = 3.85 minutes

% Idle Server = 47/115 * (100) = 41%

It is observed that two devices have been used to produce artificial (simulated) experience with a system in order to examine some of its time-dependent behavioural characteristics. The system

simulated had two random variables, the time between arrivals and the service time. Although the simulation such as this one was done by hand it becomes necessary once the system to be simulated becomes the least bit complicated or has a large number of component parts, here a computer must be used both for generating the random variables and for doing the bookkeeping.

To adequately simulate real-world systems, we must also be able to generate behavioural characteristics that are realistic. Example the time between arrivals and service times generated must allow for something other than uniform distribution rounded to the nearest whole number. Monte-Carlo sampling is a simulation prerequisite if the model is to represent stochastic or random elements realistically.

ASSIGNMENTS ONE

- A doctor's office has one nurse and one doctor.
- Patients arrive randomly. Each patient first goes through a check-up with the nurse and then proceeds for a full check-up by the doctor.
- Patients may have to wait before being served by the nurse as well as before being served by the doctor. Service priority is first-in-first-out (FIFO).
- The service time of the nurse is always 7 minutes, but the service time of the doctor is random. The distributions of the time between arrivals and the service time of the doctor are given below:

Time between arrivals [min]	Probability
5	0.5
10	0.3
15	0.2

Doctor's service time [min]	Probability
4	0.3
8	0.5
12	0.2

- Use simulation to determine the average time a patient spends in the doctor's office from the arrival till leaving.

2.12 Overview of Available Software for Simulation Modelling

Simulation can be carried out using any one of the software's available in today's market. With various software available today for simulation it is imperative that we chose the correct software.

Here is a list of available simulation software:

1. ***20-sim:*** 20-sim is relatively new software used for blocks modeling system for continuous systems. 20-sim is a modeling and simulation program that runs under Windows. With 20-sim one can simulate the behavior of dynamic systems, such as electrical, mechanical and hydraulic systems or any combination of these systems.

2. ***Arena:*** This software is used to simulate service, manufacturing, transformation, and logistics, supply chain and other systems.
3. ***Automod***: This software provides true to scale 3-D virtual reality animation, making simulation models easy to understand. It provides advanced features to allow users to simulate complex movement, such as kinematics and velocity of equipment such as robots, machine tools, transfer lines, special machinery and more.
4. ***Awesim***: Awesim provides a simulation engine focused on the production of model animations. Animations can be built graphically and the user can specify controls to build interactive simulations.
5. ***EASY5***: EASY5 developed by Boeing Inc., is software used to model and simulate dynamic systems containing hydraulic, pneumatic, mechanical, thermal, electrical and digital sub-systems. A complete set of control system modeling, analysis and design features is included.
6. ***Idef:*** This is a process mapping software, which provides easy way to capture flows and present them as flows diagrams. These can present more information than traditional flow charts. Process described, constraints affecting other processes, the role of people and other resource that are involved in the process can all be incorporated.
7. ***Intrax:*** This software can support numerous management decisions regarding the actual process being modeled and simulated. It can be used to perform strategic (reality checks on strategic vision, synchronous value chain vision), process improvement (sequence improvements, test productivity improvements, reduce cycle times), synchronous value chain (dynamic visualization, establish current constraints) and day-to-day operations (compare operational alternatives; test the impact of short-term changes).
8. ***Manufacturing engineering:*** This software performs discrete simulations to resolve manufacturing problems and design manufacturing solutions. It predicts throughput, staffing and other performance measures in a broad range of applications.
9. ***Matlab:*** This is an integrated technical computing environment that combines numeric computation, advanced graphics and visualization, and a high level programming language.

10. **Simulink** is an interactive tool for modeling, simulating and analyzing dynamic systems. It allows building graphical block diagrams, simulating dynamic systems, evaluating system performance and refining designs.
11. ***Modsim***: This software can be used to simulate transportation models like port simulation model, railroad network simulation and aircraft/air traffic management model. It is also used for manufacturing simulation.
12. ***Promodel:*** This software is simulation-based software for evaluating, planning or re-designing manufacturing, warehousing and logistics systems. Typical applications include implementation of lean manufacturing, cycle-time reduction, equipment investment decisions, throughput & capacity analysis, identifying and minimizing bottlenecks and resource allocation.
13. ***Prosolvia:*** This software brings reality to virtual manufacturing. It allows the user to view the product in a variety of views and situations. End user interaction and functional testing in the early concept stages of the product development process means that important design and manufacturing decisions can be made much earlier in the process. For the manufacturing process, staffing training for assembly and maintenance can be developed much earlier in the pre-production process.
14. ***Quest:*** This software provides the user with a graphics and visual analysis capability. It enables accurate modeling of conveyors, buffers, docks, ASRS, etc.
15. ***SDI supply chain:*** This software provides a tool for studying the impact of changing demand, logistics decisions and production policies on key system performance measures. It is used to model the dynamics of a complete supply chain from source to user, along the entire plan, source, and make and deliver process. It allows users to design, analyze and study areas such as supply chain capacity issues, bottleneck identification, logistics deployments, resource deployments, system velocities and reliability. It also models the network of suppliers, warehouses and shipping channels, which provides materials to a production plant. Supply chain performance is measured in terms of total product cost, cash-in-system, time-in-system and reliability.
16. ***Simba:*** Lanner's Simulation base application software enables the rapid development and component build of simulation embedded operational planning and scheduling applications for supply chains.

17. ***Simplorer:*** A simulator for industrial design, research projects and teaching purposes. External code can be embedded. Has its own language. Allows for hierarchical structures.
18. ***Witness (SDX):*** This software offers discrete event simulation. This software has various facilities, which enable easy simulation of automotive manufacturing. Cycle times, breakdown modes and timings, setup modes and timings, buffer capacities, buffer dwell times, machine type etc. can be easily brought out as output together with routing information. It also includes material flow optimization. Effective material flow planning minimizes the high cost of moving materials and products from one machine to another.

CHAPTER 3 DISCRETE EVENT SIMULATION (DES)

3.1 Chapter objectives and expected results

The objective of this chapter is to:

- Look at the concepts of discrete event simulation which provides an implementation framework for most simulation languages, regardless of the user worldview supported by them.

At the end of this chapter, students are expected to have a general understanding of discrete event simulation which is one of the paradigms implemented by modern computer simulation tools.

3.2 Elements of Discrete Event Simulation

In the field of simulation, a discrete-event simulation (DES), models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation can directly jump in time from one event to the next.

A system state is a set of data that captures the salient variables of the system and allows us to describe system evolution over time.

In a computer simulation program, the state is stored in one or more program variables that represent various data structures e.g., the number of customers in a queue, or their exact sequence in the queue. As an example, consider a machine, fed by raw-material storage of jobs. A "coarse" state of the system is the number jobs in the storage; note, however, that this state definition does not permit the computation of waiting times, because the identity of individual jobs is not maintained. On the other hand, the more "refined" state consisting of customer identities in a queue and associated data such as customer arrival times does permit the computation of waiting times. In practice, the state definition of a system should be determined based on its modeling needs, particularly the statistics to be computed.

The state trajectory over time, S (t), is abstracted as a step function, whose jumps (discontinuities) are triggered by discrete events, which induce state transactions (changes in the system state) at particular points in time. Although computer implementation of events varies among DES simulators, they are all conceptually similar:

An event is a data structure that always has a field containing its time of occurrence, and any number of other fields.

The evolution of any DES model is governed by a clock and a chronologically ordered event list. The event at the head of the list is called the most imminent event for obvious reasons. Scheduling an event means that the event is linked chronologically into the event list. The occurrence of an event means that the event is unlinked from the event list and executed. The execution of an event can change state variables and possibly schedule other events in the event list.

An essential feature of the DES concept is that "nothing" changes the state unless an event occurs, at which point the model typically undergoes a state transition.

More precisely, every event execution can change the state although on rare occasions the state remains intact, but every state change is effected by some event. Between events, the state of the DES is considered constant, even though the system is engaged in some activity.

For example, consider a machine on the factory floor that packages beer cans into six-packs, such that the next six-pack is loaded for processing only when the previous one has been completely processed. Suppose that the state tracks the number of six-packs waiting to be processed at any given time. Then during the processing time of a six-pack, the DES state remains unchanged, even though the machine may, in fact, be processing six beer cans individually. The DES state will only be updated when the entire six-pack is processed; this state change will be triggered by a "six-pack completion" event.

At the highest level of generalization, a DES simulator executes the following algorithm:

1. *Set the simulation clock to an initial time (usually 0), and then generate one or more initial events and schedule them.*
2. *If the event list is empty, terminate the simulation run. Otherwise, find the most imminent event and unlink it from the event list.*
3. *Advance the simulation clock to the time of the most imminent event, and execute it (the event may stop the simulation).*
4. *Loop back to Step 2.*

3.3 Examples of DES Models

The power and generality of DES models are illustrated through several examples of elementary systems. The examples illustrate how progressively complex DES models can be constructed from

simpler ones, either by introducing new modeling wrinkles that increase component complexity, or by adding components to create larger DES models.

3.3.1 Single Machine

Consider a failure-proof single machine on the shop floor, fed by a buffer. Arriving jobs that find the machine busy (processing another job) must await their turn in the buffer, and eventually are processed in their order of arrival. Such a service discipline is called FIFO (first in first out) or FCFS (first come first served), and the resulting system is called a queue or queueing system.
Suppose that job inter-arrival times and processing times are given (possibly random). A schematic description of the system is depicted in Figure 3.1.

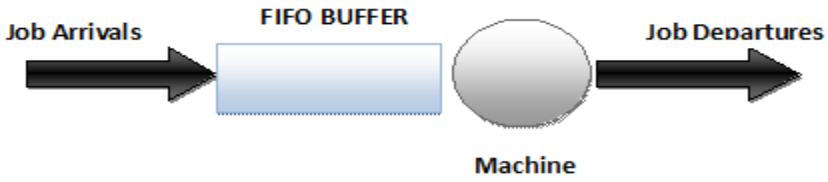


Figure 3.1 A single FIFO machine.

To represent this system as a DES, define the state S (t) to be the number of jobs in the system at time t. Thus, *S(t) = 5* means that at time t, the machine is busy processing the first job and 4 more jobs are waiting in the buffer. There are two types of events: ***arrivals*** and ***process completions***. Suppose that an arrival took place at time t, when there were *S (t) = n* jobs in the system. Then the value of *S* jumps at time *t* from *n* to *n +1*, and this transition is denoted by *n* → *n +1*. Similarly, a process completion is described by the transition *n*→ *n - 1*. Both transitions are implemented in the simulation program as part of the corresponding event processing.

3.3.2 Single Machine with failures

Consider the previous single machine on the shop floor, now subject to failures.
In addition to arrival and service processes, we now also need to describe times to failure as well as repair times. We assume that the machine fails only while processing a job, and that on repair completion, the job has to be reprocessed from scratch. A schematic description of the system is depicted in Figure 3.2.

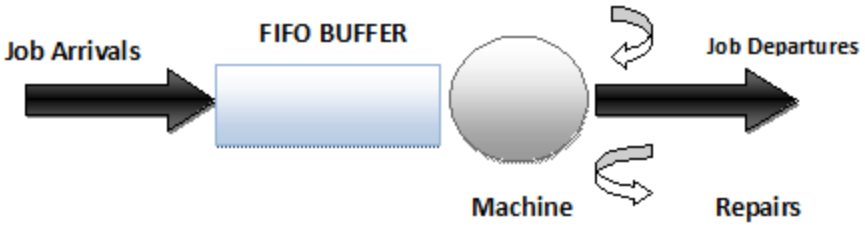


Figure 3.2 A single FIFO machine with failures.

The state *S (t)* is a pair of variables, *S (t) = (N (t), V (t))*, where *N (t)* is the number of jobs in the buffer, and *V (t)* is the process status (idle, busy, or down), all at time t. In a simulation program, *V (t)* is coded, say by integers, as follows: 0 = idle, 1 = busy and 2 = down. Note that one job must reside at the machine, whenever its status is *busy or down*. The events are *arrivals, process completions, machine failures,* and *machine repairs.*

The corresponding state transitions follow:

- *Job arrival: (n, v) →(n = 1, v)*
- *Service process completion: (n, 1) →{ (0, 0), if n = 1 (n - 1, 1), if n > 1*
- *Failure arrival: (n, 1) →(n, 2)*
- *Repair completion: (n, 2) →(n, 1)*

3.3.3 Single Machine with an Inspection Station and Associated Inventory

Consider the single machine on a shop floor, without failures.

Jobs that finish processing go to an inspection station with its own buffer, where finished jobs are checked for defects. Jobs that pass inspection are stored in a finished inventory warehouse. However, jobs that fail inspection are routed back to the tail end of the machine's buffer for reprocessing. In addition to inter-arrival times and processing times, we need here a description of the inspection time as well as the inspection decision (pass/fail) mechanism (e.g., jobs fail with some probability, independently of each other). A schematic description of the system is depicted in Figure 3.3.

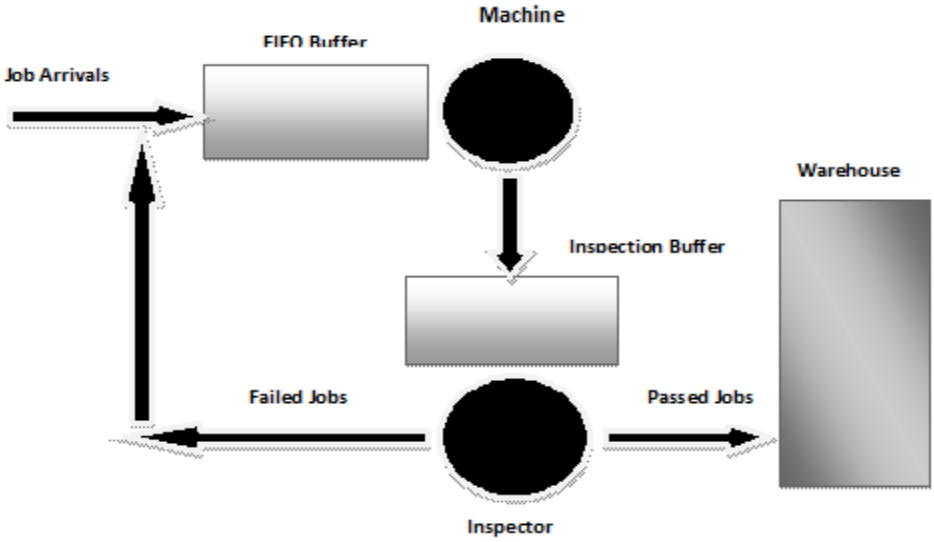


Figure 3.3 A single FIFO machine with inspection and storage.

The state *S (t)* is a triplet of variables, *S (t) = (N (t), I (t), K (t))* where *N (t)* is the number of items in the machine and its buffer, *I(t)* is the number of items at the inspection station, and *K(t)* is the storage content, all at time t. Events consist of *arrivals, process completions, inspection failure (followed by routing to the tail end of the machine's buffer),* and *inspection passing (followed by storage in the warehouse).*

The corresponding state transitions follow:

- *Job arrival: (n, i, k)* → *(n +1, i, k)*
- *Process completion: (n, i, k)* → *(n - 1, i + 1, k)*
- *Inspection completion: (n, i, k)* → *(n + 1, i - 1, k), if job failed inspection (n, i -1, k +1), if job passed inspection*

3.4 Monte Carlo Sampling and Histories

Monte Carlo simulation models incorporate randomness by sampling random values from specified distributions. The underlying algorithms and/or their code use random number generators (RNG) that produce uniformly distributed values between 0 and 1; these values are then transformed to conform to a prescribed distribution. This general sampling procedure is referred to as Monte Carlo sampling.

In particular, the random values sampled using RNGs are used to schedule events at random times. For the most part, actual event times are determined by sampling an inter-event time (e.g., inter-arrival times, times to failure, repair times, etc.) via an RNG, and then adding that value to the current clock time.

DES runs use a statistical approach to evaluating system performance; in fact, simulation-based performance evaluation can be thought of as a statistical experiment. Accordingly, the requisite performance measures of the model under study are not computed exactly, but rather, they are estimated from a set of histories. A standard statistical procedure unfolds as follows:

1. *The modeler performs multiple simulation runs of the model under study, using independent sequences of random numbers. Each run is called a* ***replication****.*
2. *One or more performance measures are computed from each replication. Examples include average waiting times in a queue, average WIP (work in process) levels, and downtime probabilities.*
3. *The performance values obtained are actually random and mutually independent, and together form a statistical sample. To obtain a more reliable estimate of the true value of each performance metric, the corresponding values are averaged and confidence intervals about them are constructed.*

CHAPTER 4 BASIC MODELING CONCEPTS

4.1 Chapter Objectives and Expected Results

The objective of this chapter is to look at the basic concepts of ARENA to construct simulation models of discrete systems. Only a small subset of ARENA's modelling features will be discussed in this chapter; however, this subset will allow us to develop and run complete models of relatively simple systems. In later chapters we will discuss additional modelling features that will greatly expand on the complexity of the systems that we can model.

At the end of this chapter, students are expected to have a general understanding of the working features and properties of the ARENA 3.0 simulation modelling software, and how to develop and run complete models of relatively simple systems.

4.2 Basics of Arena simulation modelling software

We begin looking at an overview of some general concepts, terms, and conventions employed in ARENA. Then we look at a problem, which we use as a vehicle for introducing specific modelling constructs in ARENA. After presenting the problem, we look at a small subset of ARENA features sufficient to model the problem and discuss its solution. Also we embellish the simple problem as a means of introducing some additional ARENA modelling constructs. We then conclude the introduction to ARENA software by presenting more additional examples to underscore the generality of the concepts presented.

We first look at some basic concepts and define some important terms that must be known before modelling processes in ARENA.

4.2.1 The Relationship between the Model and Experiment

In ARENA's modelling framework, there is a fundamental distinction between the Model Frame and the Experiment Frame. The model is a functional description of the systems components and their interactions. The experiment, on the other hand, defines the experimental conditions (run length, initial conditions, etc.) under which the model is expected to be exercised to generate a specific output data. An ARENA simulation program comprises both a model and a corresponding experiment. We will first concentrate on developing the model frame but in describing ARENA's

model frame, we occasionally will refer to information that is separately defined in the Experimental Frame.

4.2.2 Entities, Attributes, and Process

Discrete systems are normally modelled in ARENA by using process orientation. In a process orientation, a particular system is modelled by studying the entities that move through that system. Our model consists of a description of the processes through which the entities move as they progress through the system.

An entity is a generic term used in ARENA to denote any person, object, or thing whether real or imaginary whose movement through the system causes changes in the state of the system.

Therefore, within a particular given system, there can be many types of entities, and each can have specific, unique characteristics. These characteristics are referred to in ARENA as attributes. In a factory, for example, an entity may correspond to a work piece and have attributes specifying the part number, due date, and priority of the work piece.

In ARENA, entities are dynamic; their entrance to and exit from the model correspond to their arrival and departure from the system. The number of entities in the model changes each time a new entity enters the model or an existing entity exits the model.

The term process, as used in ARENA, denotes the sequence of operations or activities through which the entities move.

Processes are static and are activated by entities. In a factory, for example, a process may consist of s drilling operation, followed by an inspection activity. The process remains dormant unless an entity, such as work piece, arrives to activate the process.

4.2.3 Block diagram and Basic Block Types used in ARENA

Processes are modelled in ARENA by using block diagrams. A block diagram is basically a linear, top-down flow graph depicting the process through which the entities in the system move. The block diagram is constructed as a sequence of blocks, the shapes and names of which indicate their general function. The experimental section of the program consists of elements which, are specified interactively in the Arena modelling environment.

The sequencing of blocks is shown by arrows, which represent the flow of entities from block to block. A model is constructed by selecting standard blocks from the available set and combining them into block diagrams in such a way that the block diagram describes the process being modelled. The blocks are selected and arranged within the block diagram based on their functional operation and interaction.

4.2.3.1 Block Types used in ARENA

The following describes in brief the different blocks used by the simulation team in ARENA:

- ***CREATE:*** It is one of the several mechanisms by which an entity (product) can enter into the model. It is typically used to model arrival processes in which entities sequentially enter the model according to a specified pattern. The operands of Create block are Batch size indicating the number of entities arriving simultaneously in the model, Offset time specifying the time between the start of simulation and the arrival of the first entity, Interval specifying the delay between the arrivals at successive points after the first. The Maximum Batches operand specifies the maximum number of arrival points.
- ***QUEUE:*** It provides waiting space for the entities whose movements through the model have been suspended based on the system's status. An example is a work piece waiting in turn to be processed on a busy machine. The operands of queue block are Queue ID indicating the name of the queue, Capacity of the queue and the Balk label which is used to direct the entity to an alternative block other than the seize block.
- ***SEIZE:*** It is used in conjunction with the queue block and is used to model the status delays. When a resource becomes idle the entity from the previous queue block enters the seize block and seizes the resource. The state of the resource now changes from idle to busy. The operands of seize block include priority for the allocation of entities waiting for the same resource. Priority is given to those entities, which have been waiting the longest. The Resource ID indicating the resource requested by the waiting entity and the number of units of the resource requested by the entity.
- ***DELAY:*** Once an entity has been allocated the necessary resources, it typically engages in time-consuming activities, such as setup, machining, inspection etc. In ARENA3.0, delays such as these can be modeled by using the DELAY block. The operands of the delay block

are the duration of the delay and the storage ID, which provides statistics on the number of entities residing in one or more DELAY blocks.

- ***RELEASE***: When an activity that requires resources has been completed, the entity possessing the resources typically releases them so that they can be allocated to entities either currently waiting or yet to arrive at QUEUE-SEIZE blocks. Operands of seize block are Resource ID indicating the resource just released and the Quantity of entities to release simultaneously.
- ***COUNT:*** It is used to count the number of occurrences of some event, e.g., work-pieces entering the system, exiting the system or sent through rework. The operands of the Count block include the Counter ID (name of the counter) and the Counter increment for which, a default value of one is assumed.
- ***DISPOSE:*** It provides a mechanism for modeling the departure of entities from the system. The dispose block has a single entry point, no operands, and no exit points. All entities entering the dispose block are terminated from the model.
- ***GROUP:*** It is commonly used to group a set of entities having a particular set characteristic. A representative entity is created and it moves through the model and behaves, as though, it is a single entity. The most important operand of group block is the quantity to group. This is commonly used to simulate a manufacturing scenario in which a set of parts corresponding to a particular product need to be grouped before they can be assembled by the machine.
- ***SPLIT:*** The entities, which had been grouped, need to be split or de-grouped before they are disposed. The split block is used for the same. The departing entities have the original attribute values they had when the entity set was formed.
- ***ASSIGN:*** Whenever attributes and variables are used in a model they need to be assigned values during model execution. This is made possible by means of the assign block. The operand of the block is Variable or Attribute = Value. Each time an entity passes through the ASSIGN block, the value on the right hand side of the equals sign is copied on to the variable or attribute on the left hand side of the equation. Example when the entities pass through the CREATE block they can be assigned a job type of 1, 2, 3 with a certain probability of the entity belonging to each type.

- ***BRANCH:*** This is used to direct the entities to different sections of the model depending upon the condition of true or false. The operands of the branch block are the following:
    - Maximum Number of Branches (it is usually 1).
    - With (indicates the probability and the label to which the entity needs to be directed).
    - Else (indicates the label to which the entity needs to be directed if the else condition becomes true).
    - Always (whatever the condition the entity needs to be sent to a specific label).

As described earlier, experiment defines the experimental conditions under which the model is exercised to generate specific output data. The length of the simulation run, the number of replications of the simulation, the characteristics of resources and queues, etc all comes under experimental conditions. Using special data records called elements the experiment is developed. It is generally preferable to enter the experiment before entering the model. By first defining the objects such as resources in the experiment, the drop down list can then be selected when entering the model graphically.

Elements

The following describes in brief the various elements used by the simulation team in developing the project.

- ***PROJECT:*** It is used to describe the simulation project used by ARENA3.0 in labeling the SIMAN summary report. The program automatically generates this report at the end of each replication if it is defined under the experimental conditions. The operands of the element include Project Title, Analyst Name, Date, Summary Report (yes/ no). Maximum of 24 characters can be entered in the first two fields.
- ***QUEUES:*** It defines the information of the model's queues. The operands of the queue element are Number (need not be defined), Name of the queue, Ranking Criterion. The following criterions are used:
    - FIFO (First-in, First-out): Entities wait in the queue in the order in which they arrive.

- LIFO (Last-in, First-out): Entities are ranked in the reverse order of their entry into the queue.
- LVF (Attribute ID): The queue is ordered by increasing values of the specified attribute with ties broken by the FIFO rule.
- HVF (Attribute ID): Queue is ordered by the decreasing values of the specified attribute with ties broken by the FIFO rule.

- ***RESOURCES:*** It defines the information about the model's resources. The operands include Number (need not be defined), Name of the resource, Capacity, which defines the number of identical and interchangeable units that initially exist for the resource. It defaults to 1.
- ***COUNTERS:*** This element provides descriptive information of the counters included in the model section. The operands included in the element are Number (need not be defined), Name of the counter, Limit. The last operand specifies a certain value which when exceeded causes the program to terminate.
- ***REPLICATE:*** It defines the Number of Replications for which the simulation is to be run, the Beginning Time of the first replication and the Replication Length, which defines the duration of each simulation replication.
- ***ATTRIBUTES:*** and ***VARIABLES:*** These elements provide general information about the symbolic names and properties of the attributes and variables. The operands of the elements include Number (optional), Name (index), Initial Values. The second operand is used to specify an array with certain index values. Each element in the array has a unique index value, which is stored in a common array name. The third variable is used to initialize a variable to a certain value.
- ***STATIONS:*** It defines the information of the stations used in the model. The operands include Number (optional), Name of the station.
- ***SETS:*** It defines a group of similar elements that may be referenced by a common name and set index. The elements that make up the set are referred to as the members of the set. Typical sets would contain groups of resources, queues, stations, pictures, counters, tallies, expressions, etc. The operands of the element are Number (optional), Name of the Set and the Members of the set. The last operand can be referenced by the set name. The function used by the Simulation team in building the manufacturing model was MemIdx (Setname,

Member Name). This function returns the index number of a specific member in the set. The member name was the first station at which the entity entered the model.

- ***SCHEDULES:*** This is used to change the capacity of the resource over time. The resource follows a time dependent schedule as specified in the element. For example, the resource is idle for the first 30 minutes of the simulation run. During this period it has a capacity of 0. During the next 30 minutes of the run the resource can have a capacity of 1. The operands of the element include Identifier, which is fed into the resource block following the schedule and Resource Capacity and the Capacity Duration, which specifies the capacity of the resource during a certain time period.
- ***SEQUENCES:*** It defines the sequence in which an entity visits the stations. The operands include Number (optional), Name of the sequence, Station ID and the Assignments (duration of delay, assignment of picture or job type, etc.) at each station.

4.3 The ARENA Window

After Arena starts, the computer screen shows the Arena window. Figure 4.1 shows what the Arena window typically looks like. This is the academic version 13.00.00

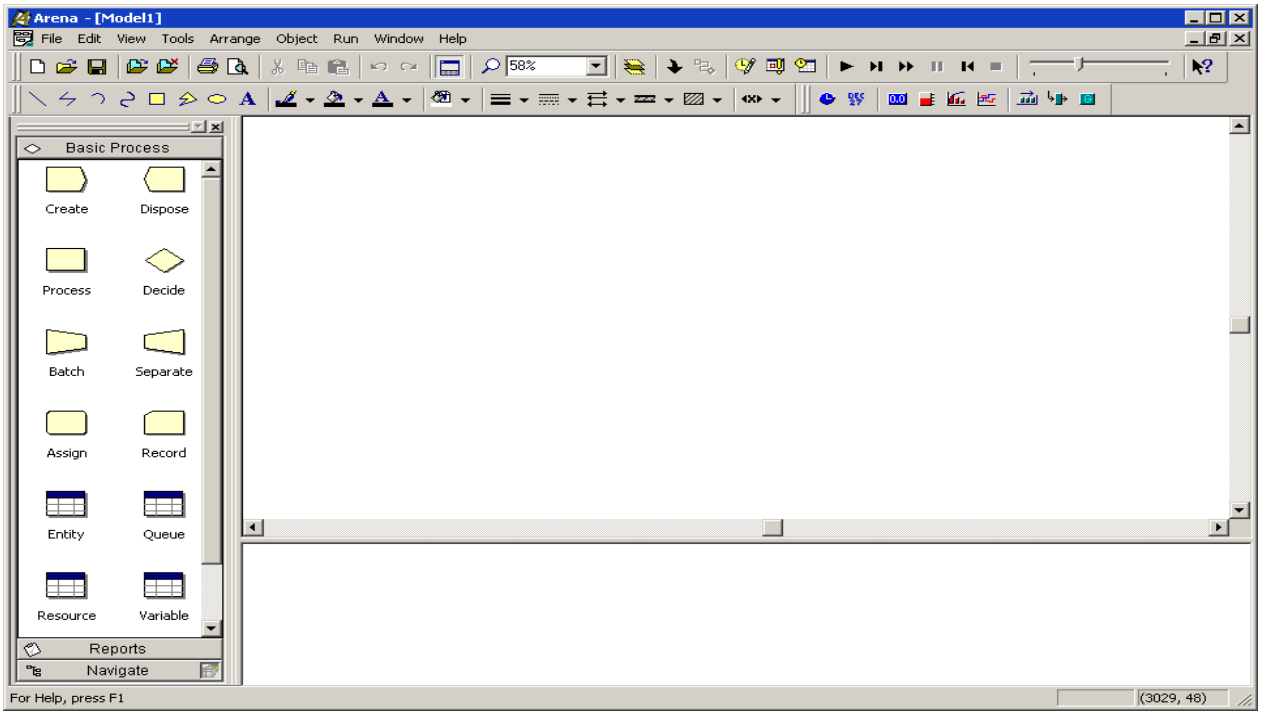


Figure 4.1 Arena Window

The Arena home screen has a ***Title*** bar with the model name at its top. Below the Title bar is the ***Arena Menu*** bar, which consists of a set of general menus and Arena-specific menus. Below the ***Menu*** bar is a set of Arena toolbars that can be displayed and hidden by clicking the right mouse button on the background area of a specific toolbar. These toolbars consist of buttons that support model building and running; note that some toolbar buttons conveniently duplicate the functionality of certain menu options in the ***Menu*** bar. The bulk of the home screen is allocated to a model window canvas consisting of a ***flowchart view*** and ***spreadsheet view***. The user spawns modules and other objects, drags them into the flowchart view canvas, and progressively builds a model from component modules selected from the ***Project bar*** (on the left) with the aid of support functions. Selecting a module in the flowchart view pops up a spreadsheet view (below the flowchart view) that summarizes all module information and permits editing of the associated data. In addition to the ***Project bar***, the user can also pop up (or hide) two additional bars below the model window canvas. The ***Debug bar*** permits the user to track the evolution of simulation runs, while the ***Status bar*** (below it) displays instructions and feedback messages concerning user actions.

4.3.1 MENU BAR

The Arena Menu bar consists of a number of general menus - File, Edit, View, Window, and Help - which support quite generic functionality. It also has the following set of Arena-specific menus:

- *The Tools menu provides access to simulation related tools and Arena parameters.*
- *The Arrange menu supports flowcharting and drawing operations.*
- *The Object menu supports module connections and sub-model creation.*
- *The Run menu provides simulation run control. Its Setup. . . option opens a form that permits the user to enter information, such as project parameters (name, analyst, date, etc.), as well as replication parameters (length, warm-up period, etc.). It also has options that provide VCR-type functionality to run simulation replications and run control options to monitor entity motion, variable assignments, and so on.*

4.3.2 Project Bar

The Project bar lets the user access Arena template panels, where Arena modules, SIMAN blocks, and various other objects cohabit. Template panels can be attached to the *Project bar* by clicking the *Attach button* on the *Standard toolbar*. More specifically, when the Attach button is clicked, a dialog box pops up on the screen and shows the so-called *.tpo* files corresponding to each template panel. Choosing a *.tpo* file will attach its template panel to the Project bar. The Arena template panels available to users are as follows:

- The ***Basic Process template*** panel consists of a set of basic modules, such as *Create, Dispose, Process, Decide, Batch, Separate, Assign,* and *Record.*
- The ***Advanced Process template*** panel provides additional basic modules as well as more advanced ones, such as *Pickup, Drop-off,* and *Match*.
- The ***Advanced Transfer template*** panel consists of modules that support entity transfers in the model. These may be ordinary transfers or transfers using material-handling equipment.
- The ***Reports template*** panel supports report generation related to various components in a model, such as entities, resources, queues, and so on.
- The ***Blocks template*** panel contains the entire set of SIMAN blocks.
- The ***Elements template*** panel contains elements needed to declare model resources, queues, variables, attributes, and some statistics collection.

In addition to the Arena template panels above, the following Arena template panels from earlier versions are also supported:

- The ***Common template*** panel contains Arena modules such as *Arrive, Server, Depart, Inspect,* and so on, as well as element modules such as *Stats, Variables, Expressions,* and *Simulate.*
- The ***Support template*** panel contains a subset of frequently used SIMAN blocks.

4.3.3 Standard Tool Bar

The *Arena Standard toolbar* contains buttons that support model building. An important button in this bar is the *Connect button*, which supports visual programming. This button is used to connect Arena modules as well as SIMAN blocks, and the resulting diagram describes the flow of logical control. The *Time Patterns Editor feature* consists of three buttons that allow the modeler to

schedule the availability of resources and their service rate. The Standard toolbar also provides VCR-style buttons to run an Arena model in interrupt mode to trace its evolution.

4.3.4 Draw and View Bar

The *Draw toolbar* supports static drawing and coloring of Arena models. In a similar vein, the *View toolbar* assists the user in viewing a model. Its buttons include *Zoom In, Zoom Out, View All,* and *View Previous*. These functions make it convenient to view large models at various levels of detail.

4.3.5 Animate and Animate Transfer Bar

The *Animate toolbar* is used for animation (dynamic visualization) of Arena model objects during simulation runs. Animated objects include the *simulation clock, queues, monitoring windows for variables, dynamic plots,* and *histogram functions*. The *Animate Transfer toolbar* is used to animate entity transfer activities, including materials handling.

4.3.6 Run Interaction Bar

The *Run Interaction toolbar* supports run control functions to monitor simulation runs, such as access to SIMAN code and model debugging facilities. It also supports model visualization, such as the *Animate Connectors button* that switches on and off entity traffic animation over module connections.

4.3.7 Integration Bar

The Integration toolbar supports data transfer (import and export) to other applications. It also permits Visual Basic programming and design.

4.3.8 Debug Bar

The *Debug toolbar* supports debugging of Arena models by monitoring and controlling the execution of a simulation run. It consists of two sub-windows. The left sub-window can be used in command mode to set breakpoints, assign variable values, observe watched variables, and trace entity flows among modules. The right sub-window has tabs for viewing future SIMAN events in the model, displaying attributes of the current active entity, and watching user-defined Arena expressions.

4.4 Building a Model Using Arena

To build a simulation model and to carry out simulation runs with Arena, a user performs the following steps:

1. *Construction of a basic model* - Arena provides the model window flowchart view, which is a flowchart-style environment for building a model. The user selects and drags the flowchart module shapes into the model window and connects them to define process flow of the model.
2. *Adding data to the model parameters* - The user adds actual data (e.g., processing times, resource demands, others) to the model. This is done by double-clicking on module icons and adding data.
3. *Performing a simulation runs of the model* - The user runs the simulation and examines the results.
4. *Analysis of the simulation results provided by the automatic reports of Arena* - The user can expand the statistics.
5. *Modifying and enhancing the model according to the user needs.*

4.4.1 EXAMPLE: A simple work station

Consider a single workstation consisting of a machine with an infinite buffer in front of it. Jobs arrive randomly and wait in the buffer while the machine is busy. Eventually they are processed by the machine and leave the system. Job inter-arrival times are exponentially distributed with a mean of 30 minutes, while job processing times are exponentially distributed with a mean of 24 minutes. This system is known in queueing theory as the M/M/1 queue.

This example will compare the simulation statistics to their theoretical counterparts to gauge the accuracy of simulation results. Specifically, we shall estimate by an Arena simulation the average job delay in the buffer, the average number of jobs in the buffer, and machine utilization.

Simulating the above workstation calls for the following actions:

1. *Jobs are created, one at a time, according to the prescribed inter-arrival distribution. Arriving jobs are dispatched to the workstation.*
2. *If the machine is busy processing another job, then the arriving job is queued in the buffer.*

3. *When a job advances to the head of the buffer, it seizes the machine for processing once it becomes available, and holds it for a time period sampled from the prescribed processing-time distribution.*
4. *On process completion, the job departs the machine and is removed from the system but not before its contribution to the statistics of the requisite performance measures are computed.*

A simple approach to modeling the workstation under study is depicted in Figure 4.2.

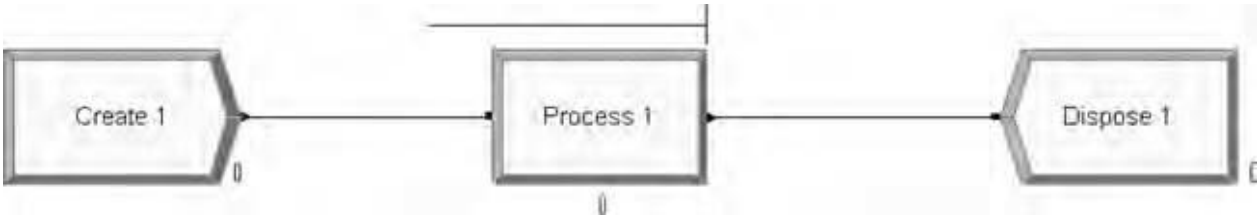


Figure 4.2 A simple Arena model of an M/M/1 queue.

In this model, jobs (Arena entities) are created by the ***Create module Create 1***. Jobs then proceed to be processed in the ***Process module Process 1***, after which they enter the ***Dispose module, Dispose 1***, for removal from the model. The graphic shaped like an elongated T (called a T-bar) above the module Process 1 represents space for waiting jobs (here, the workstation's buffer). The inter-arrival specification of the Create module is shown in the dialog box of Figure 4.3.

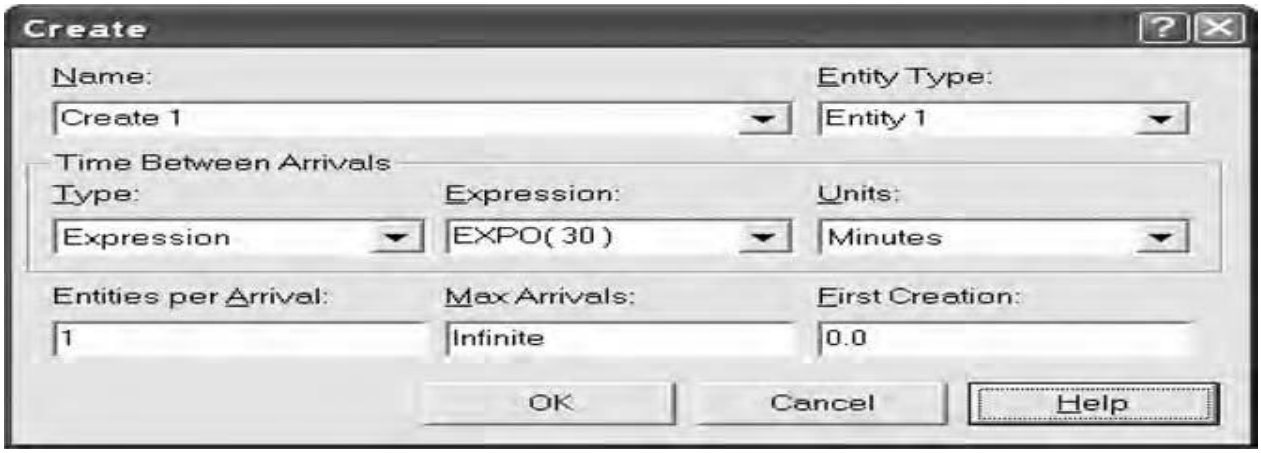


Figure 4.3 Dialog box for a Create module

The dialog box contains information on job inter-arrival time (*Time Between Arrivals section*), batch size (*Entities per Arrival field*), maximal number of job arrivals (*Max Arrivals field*), time

of first job creation (*First Creation field*), and so on. The *Type pull-down menu* in the *Time Between Arrivals* section offers the following options:

- *Random* (exponential inter-arrival times with mean given in the Value field)
- *Schedule* (allows the user to create arrival schedules using the Schedule module from the Basic Process template panel)
- *Constant* (specifies fixed inter-arrival times)
- *Expression* (any type of inter-arrival time pattern specified by an Arena expression, including Arena distributions)

The job processing mechanism (including priorities) is specified in the *Process module*, whose dialog box is shown in Figure 4.4.

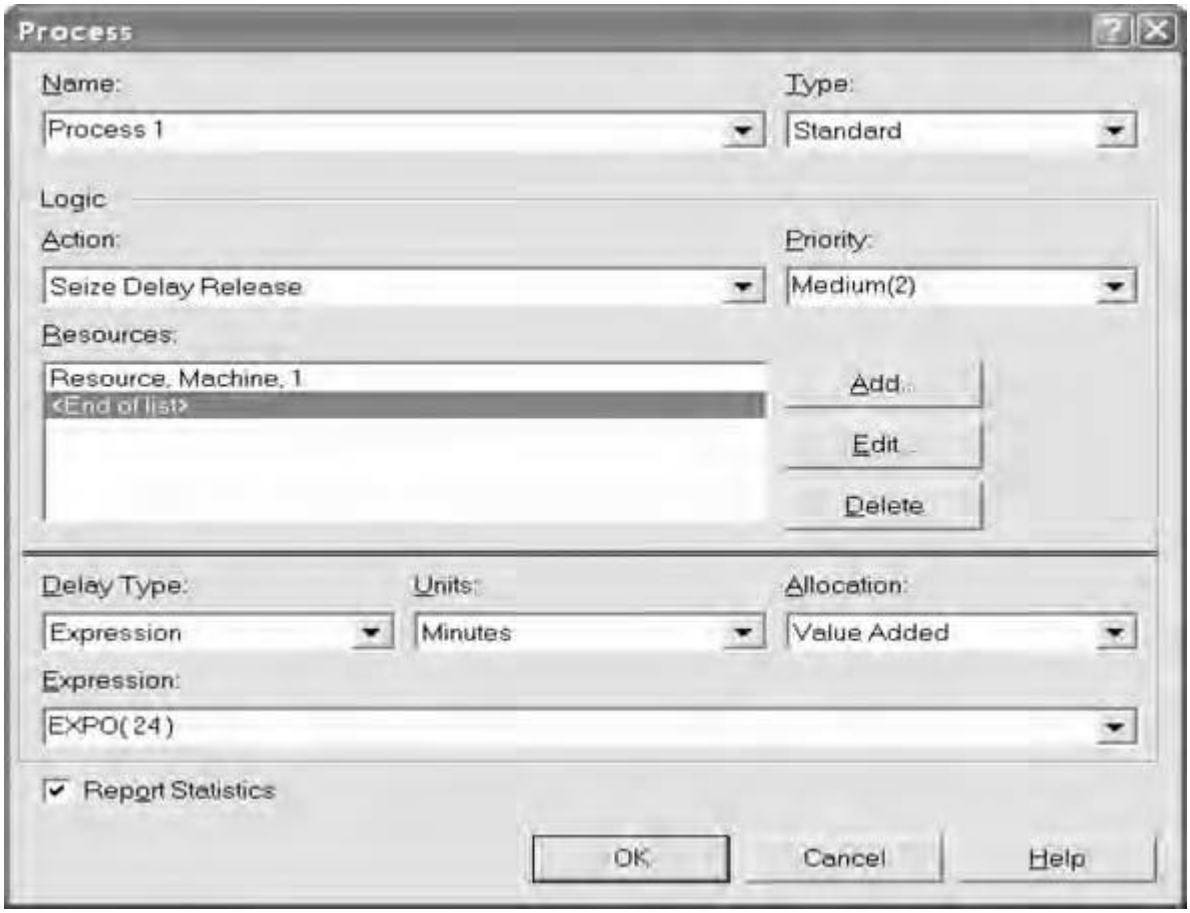


Figure 4.4 Dialog box for a Process module.

The *Action field* option, selected from the pull-down menu, is ***Seize Delay Release***, which stands for a sequence of *SEIZE, DELAY, and RELEASE SIMAN* blocks. *SEIZE* and *RELEASE* blocks are used to model contention for a resource possessing a capacity (e.g., machines). When resource

capacity is exhausted, the entities contending for the resource must wait until the resource is released. Thus the *SEIZE* block operates like a gate between entities and a resource. When the requisite quantity of resource becomes available, the gate opens and lets an entity seize the resource; otherwise, the gate bars the entity from seizing the resource until the requisite quantity becomes available. Note that the resource quantity seized should be an integer; otherwise Arena truncates it. The processing (holding) time of a resource (Machine in our case) by an entity is specified via the *DELAY* block within the Process module. For instance, the dialog box of Figure 3.4 specifies that one unit of resource Machine be seized and held for an exponentially distributed time with a mean of 24 minutes. If multiple resources need to be seized simultaneously, all requisite resources are listed, and the holding time clock is started only after all requisite resources listed become available. When at least one of the listed resources is not available, arriving entities wait in a FIFO (First In First Out) queue, ordered further by priorities (specified in the Priority field of the dialog box). This discipline is called FIFO within priority classes, because higher-priority entities precede lower-priority ones, but arrivals within a given priority class are ordered FIFO.

The *Add button* in a *Process module* (see Figure 3.4) is used to specify resources and the resource quantities to be seized by an incoming entity. To this end, the *Add button* pops up a *Resources dialog box* as shown in Figure 4.5.

Figure 4.5 Resources dialog box for specifying a resource.

Note, however, that the capacity of resources introduced in a Resources dialog box is specified elsewhere, namely, in the Resource module spreadsheet (Figure 4.6).

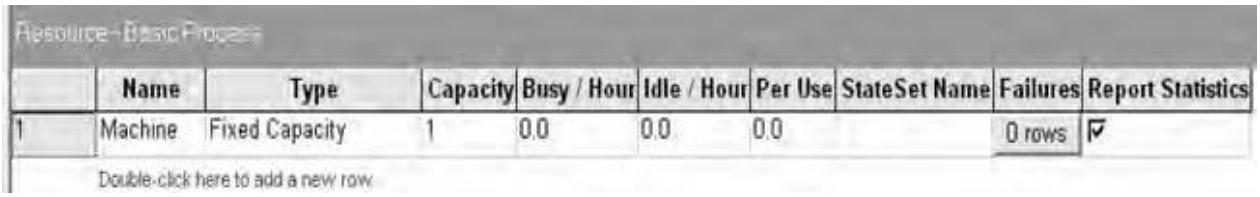


Figure 4.6 Resource module spreadsheet for specifying resource capacities.

This spreadsheet is accessible in the Basic Process template of the Project bar, and is used to specify all resource capacities of an Arena model. The default capacity of resources is 1.

The ***Dispose module*** implements an entity "sunset" mechanism, by simply discarding entities that enter it. As such, the *Dispose module* serves as a system "sink," thereby counteracting the *Create module*, which serves as a system "source." Note carefully that in the absence of properly placed *Dispose modules*, the number of entities created in the course of a simulation run can grow without bound, eventually exhausting available computer memory (or a prescribed limit) and terminating the run (very likely crashing the simulation program). A *Dispose module* is not necessary, however, when a fixed or bounded number of entities are created and circulate in the system indefinitely. Whenever an Arena model is saved, the model is placed in a file with a ***.doe*** extension (e.g., mymodel.doe). Furthermore, whenever a model, such as *mymodel.doe*, is checked using the *Check Model option* in the *Run menu* or any run option in it, Arena automatically creates a number of files for internal use, including ***mymodel.p*** (program file), ***mymodel.mdb*** (Access database file), ***mymodel.err*** (errors file), ***mymodel.opw*** (model components file), and ***mymodel.out*** (SIMAN output report file). As these are internal Arena files, the modeler should not attempt to modify them.

Run control functionality is provided by the *Run pull-down menu* (see Figure 4.1). Selecting the Setup. . . option opens the *Run Setup dialog box*, which consists of multiple tabs, each with its own dialog box. In particular, the ***Replication Parameters tab*** permits the specification of the number of replications, replication length, and the warm-up period. In this example, the model will be simulated for 10,000 hours, with all other fields in the *Run Setup dialog box* retaining their default values. The end result of a simulation run is a set of requisite statistics, such as mean waiting times, buffer size probabilities, and so on. These will be referred to as run results. Arena provides a considerable number of default statistics in a report, automatically generated at the end of a simulation run. Additional statistics can be obtained by adding statistics collection modules to the model, such as ***Record*** (Basic Process template panel) and ***Statistic*** (Advanced Process template

panel). The simpler statistics-collection facilities in Arena are one of the advantages of Arena over SIMAN. The run results of a single replication of the workstation model from Figure 3.2 are displayed in Figures 4.7 and 4.8.

3:33:44PM **Resources** September 1, 2005

A Simple Workstation Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 600,000.00 Time Units: Minutes

Machine

Usage	Value			
Total Number Seized	20,005.00			
Scheduled Utilization	0.8097			
Number Scheduled	1.0000	(Insufficient)	1.0000	1.0000
Number Busy	0.8097	0.014618405	0	1.0000
Instantaneous Utilization	0.8097	0.014618405	0	1.0000

Figure 4.7 Resource statistics from a single replication of the simple workstation model

Figure 4.7 displays the ***Resources*** section, which includes resource utilization. The last three columns correspond to the half-width of the confidence interval, minimum observation, and maximum observation, respectively, of the corresponding statistics. The ***(insufficient)*** notation indicates that the number of observations is insufficient for adequate statistical confidence. Observe that the ***Number Busy*** item refers to the number of busy units of a resource, while the ***Number Scheduled*** item refers to resource capacity. The ***Instantaneous Utilization*** item pertains to utilization per resource unit, namely, ***Number Busy*** divided by the ***Number Scheduled***. We point out that in the long run, individual resource utilizations approach this number.

3:33:07PM **Queues** September 1, 2005

A Simple Workstation Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 600,000.00 Time Units: Minutes

Process 1.Queue

Time	Average	Half Width	Minimum	Maximum
Waiting Time	107.09	16.42523	0	807.06
Other	**Average**	**Half Width**	**Minimum**	**Maximum**
Number Waiting	3.5721	0.538999733	0	35.0000

Figure 4.8 Queue statistics from a single replication of the simple workstation model.

Figure 4.8 displays the ***Queues*** section, which consists of customer-oriented statistics (customer averages), such as mean waiting times in queues, as well as queue-oriented statistics, such as time averages of queue sizes (occupancies). Additional sections in an output report include ***Frequencies*** (probability estimates) and ***User Specified*** (any customized statistics). An examination of the run results (replication statistics) displayed in Figures 4.7 and 4.8 reveals that the machine utilization is about 81%. The average waiting time in the machine buffer is about 107 minutes, with a 95% confidence interval of 107.09_16.42523 minutes, and a maximum of 807.06 minutes. The average number of jobs in the buffer is 3.57 jobs, with the maximal buffer length observed being 35 jobs. Figure 3.7 can be used for preliminary verification of the workstation model. For example, we can use the classical formula r ¼ l=m of machine utilization to verify that the observed server utilization, r, is indeed the ratio of the job arrival rate (l ¼ 1=30) and the machine-processing rate (m ¼ 1=24). Indeed, this ratio is 0.8, which is in close agreement with the run result, 0.8097. Note that this relation holds only approximately, due to the inherent variation in sampling simulation statistics.

4.4.2 EXAMPLE 2: Simple Model of the Car Wash System

Vehicles arrive into a carwash shop to get a simple wash and clean up. The Carwash system consists of a single wash machine, which provides the actual service to the vehicles. Arriving vehicles join a line to wait for service. The vehicle at the head of the line is the one that is next to be serviced by the carwash machine. After the vehicle wash is completed, the vehicle leaves the

system. The vehicles are considered the customers of the system, as they are the entities requesting service by the wash machine. Figure 4.9 shows a graphical view of the conceptual model of the Carwash system.

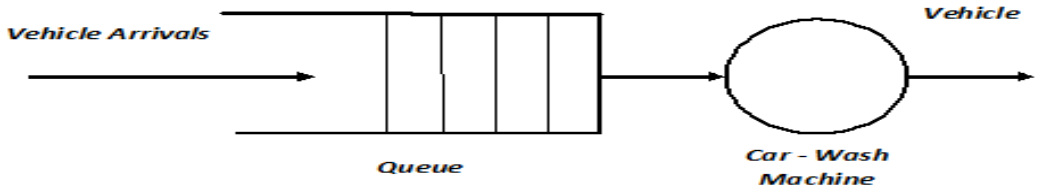


Figure 4.9 Graphical view of the conceptual model of the Carwash system

The objects that flow through the system are the vehicles and in Arena, these objects are known as *entities*. The first task in building the model is to define the flowchart of the model. To start the construction of a new model in Arena, the user activates the File menu and selects New. A new model is given the name *Model1* by default.

From the Project bar on the left-hand side of the Arena window, the user activates the Basic Process panel and drags the Create module into the flowchart view of the model window. Then the user drops the module shape in a convenient place, on the upper left-hand side of the flowchart view. Figure 4.10 shows the Create module in the flowchart view of the model window. This is the first module in constructing the model for the Carwash system. The figure shows only part of the Arena screen. The Create module is the source of arriving vehicles into the system.

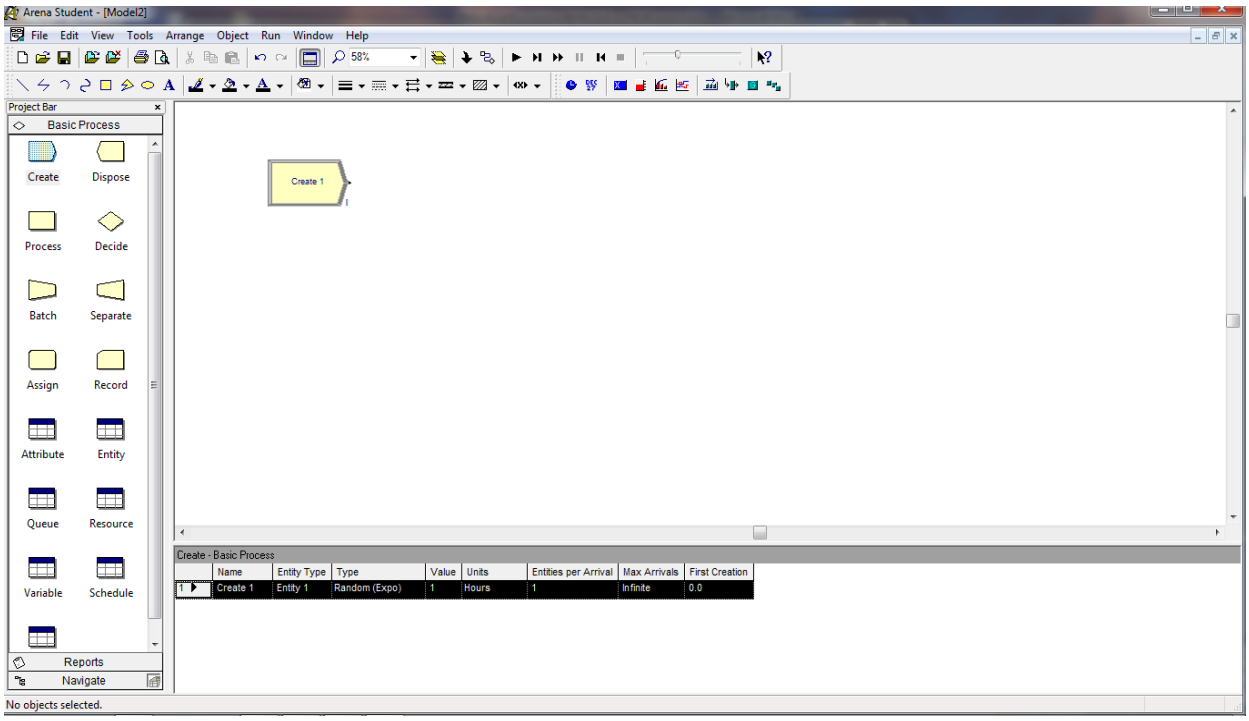


Figure 4.10 The Create module in model of the Carwash system

To assign value to attributes or properties of the module, the user double-clicks on the module shape and enters the data requested in the small dialog window, as shown in Figure4.11.

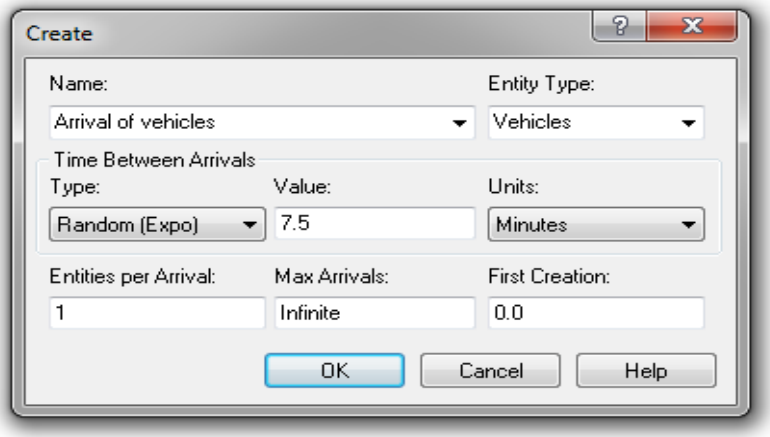


Figure 4.11 Data in the properties of the Create module

The name assigned to the module is *Arrival of vehicles*. The entity type is *Vehicles*. The arrival of vehicles occurs randomly and the time between arrivals (the inter-arrival intervals) follow the behavior represented by an exponential probability distribution with mean value of 7:5 in minutes. The data for the module could also be filled in the spreadsheet view, below the model workspace.

The other modules needed to construct this simple simulation model are:

- *A Process module and*
- *A Dispose module*

As mentioned previously, these modules must be connected in such a manner as to represent the flow of entities through the system. To place the two remaining modules in the model, the user selects the Create module, which is on the model view. The user then drags the Process module in the Basic process panel on the project bar to the flowchart model view and places it at any location to the right of the Create module. The two modules will automatically be connected. Now the user selects the Process module on the model view and drags the Dispose module in the Basic process panel on the project bar to the flowchart model view and places it at any location to the right of the Process module. Figure 4.12 shows the complete flowchart of the Carwash model. The second module in the model is a Process module with assigned name *Wash station*. The third module is a Dispose module with assigned name *Vehicles exit*.

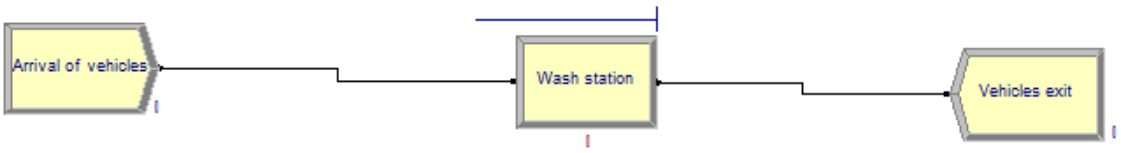


Fig. 4.12 Flowchart of the Carwash model

The properties of the Process module were assigned in a similar manner as for the Create module. The user double-clicks on the Process module and opens the dialog window of the module. The actual values to be used are then assigned to all the properties of the module, as shown in Figure 4.13.

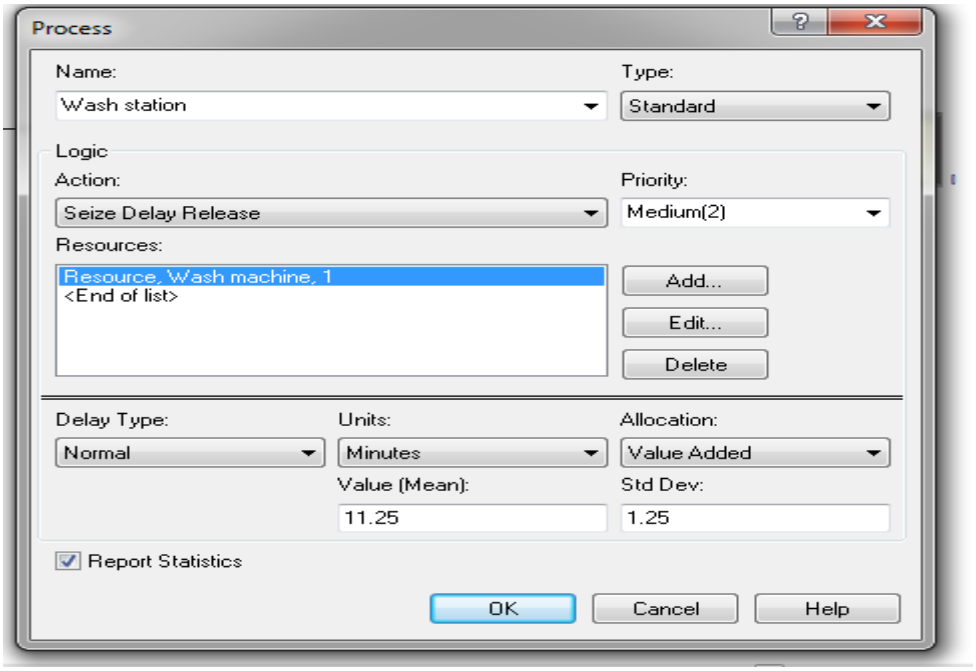


Figure 4.13 Properties of the Process module of the Carwash model

The name assigned to the Process module is *Wash station*. The Action selected for the module is *Seize, Delay, Release*. This means that a vehicle that arrives will wait until the resource becomes available, it will seize the resource, it will wait for the service interval, and then it will release the resource. The *Delay* is actually the processing time or the service interval. To specify the resource that the vehicle needs for service, the user clicks the Add button on the Resources area of the Process module. A small Resources dialog window appears. The resource is specified and the name assigned to it is *Wash machine*. Figure 4.14 shows the Resources dialog window.

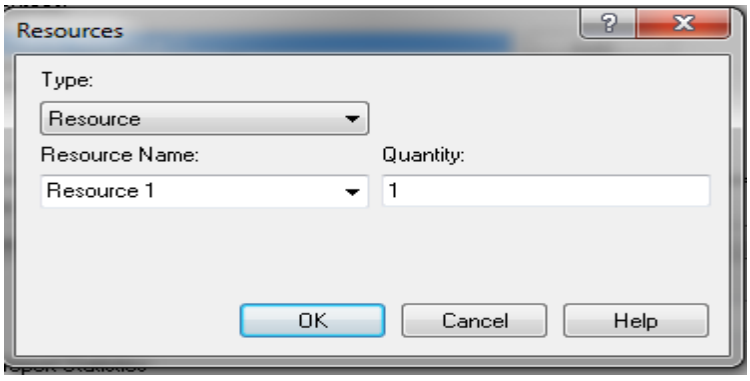


Figure 4.14 The resources dialog window in the Carwash model

To setup the simulation runs, the user selects the Run menu and selects Setup. The dialog window that appears is shown in Figure 4.15. This figure shows the project title in the Project Parameters tab.

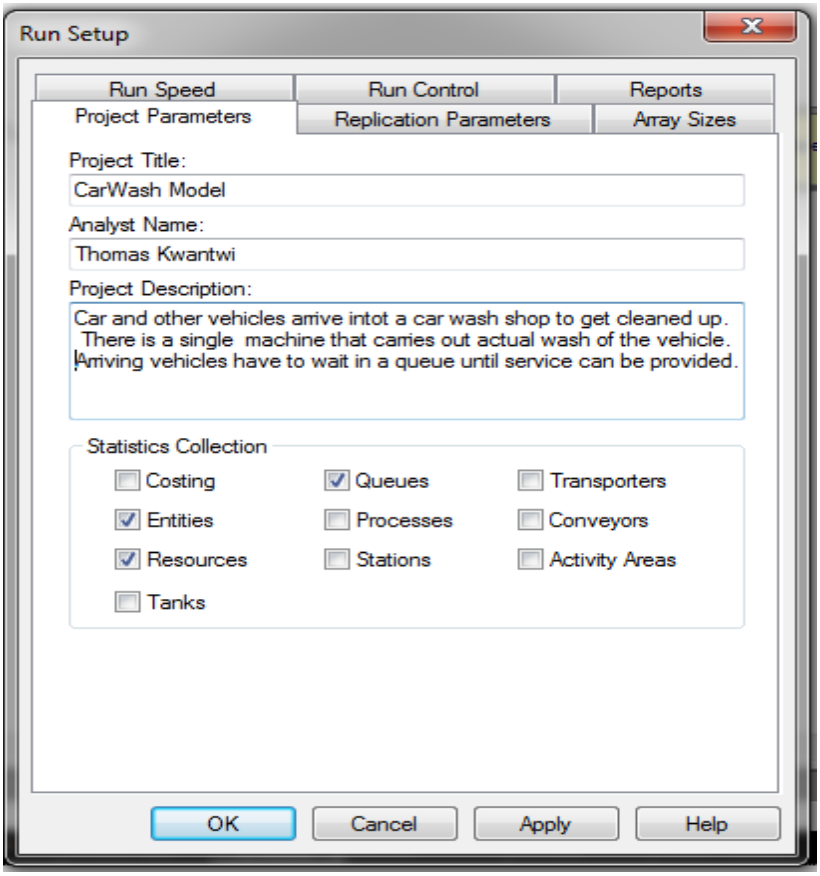


Figure 4.16 The Project Parameters in the Run Setup dialog window of the Carwash model

The Replications Parameters tab of the Run Setup dialog window is shown in Figure 4.17. The main parameter set is the replication Length, which is the simulation period.

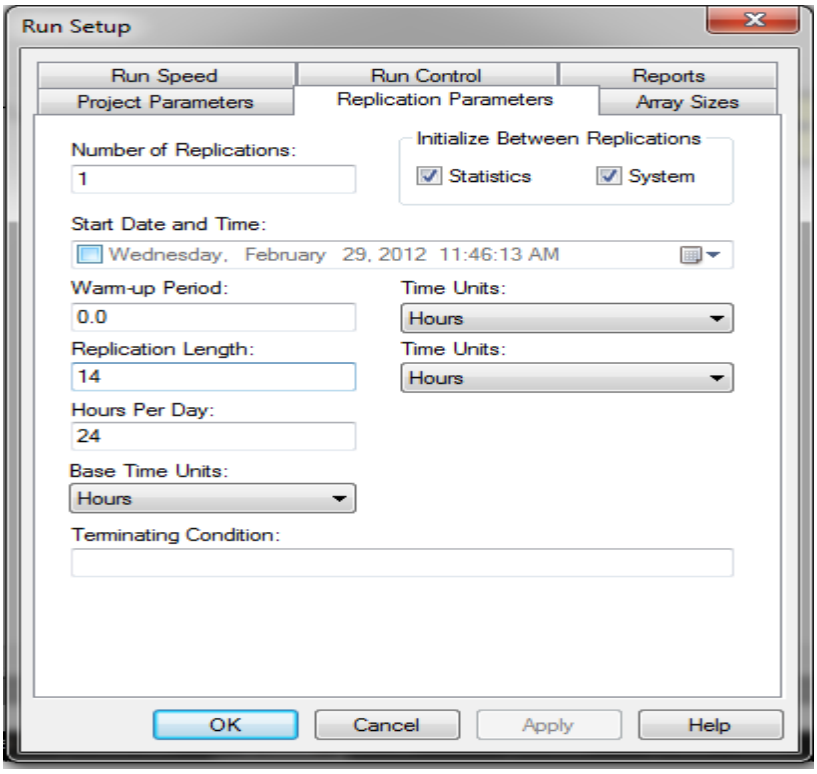


Figure 4.17 The Replication Parameters on the Run Setup dialog window of the Carwash model

To run the simulation, the user selects Go from the Run menu, or presses F5. After the simulation runs to completion, Arena will ask the user whether he/she needs to open the reports. The various reports produced by Arena after a simulation run can be accessed and opened from the Reports panel in the Project bar. Arena opens the Category Overview Report by default. Figure 4.18 shows part of the Category by Replications report. There is also a text file with a summary of results in the file Carwash out.

11:54:39AM **Category by Replication** February 29, 2012

CarWash Model Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 14.00 Time Units: Hours

Entity

Time

VA Time	Average	Half Width	Minimum	Maximum
Vehicles	0.1923	(Insufficient)	0.1391	0.2401
NVA Time	**Average**	**Half Width**	**Minimum**	**Maximum**
Vehicles	0	(Insufficient)	0	0
Wait Time	**Average**	**Half Width**	**Minimum**	**Maximum**
Vehicles	2.2082	(Insufficient)	0	4.4362
Transfer Time	**Average**	**Half Width**	**Minimum**	**Maximum**
Vehicles	0	(Insufficient)	0	0
Other Time	**Average**	**Half Width**	**Minimum**	**Maximum**

Figure 4.18 The Category by Replication report of the Carwash model

From this report, the average wait time of a vehicle is 2.2082 minutes. The maximum wait time is 4.4362 minutes. The average time that a vehicle spent in the car wash shop is 2.4005 minutes. The total number of vehicles that arrived is 106 and the total number of vehicles that were serviced is 106, the rest remained in the system.

Figure 4.19 shows part of the Resources report produced after the simulation run. The resource (wash machine) *utilization* was 100%. The total number of times that the resource was seized was 73.

11:58:57AM **Resources** February 29, 2012

CarWash Model Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 14.00 Time Units: Hours

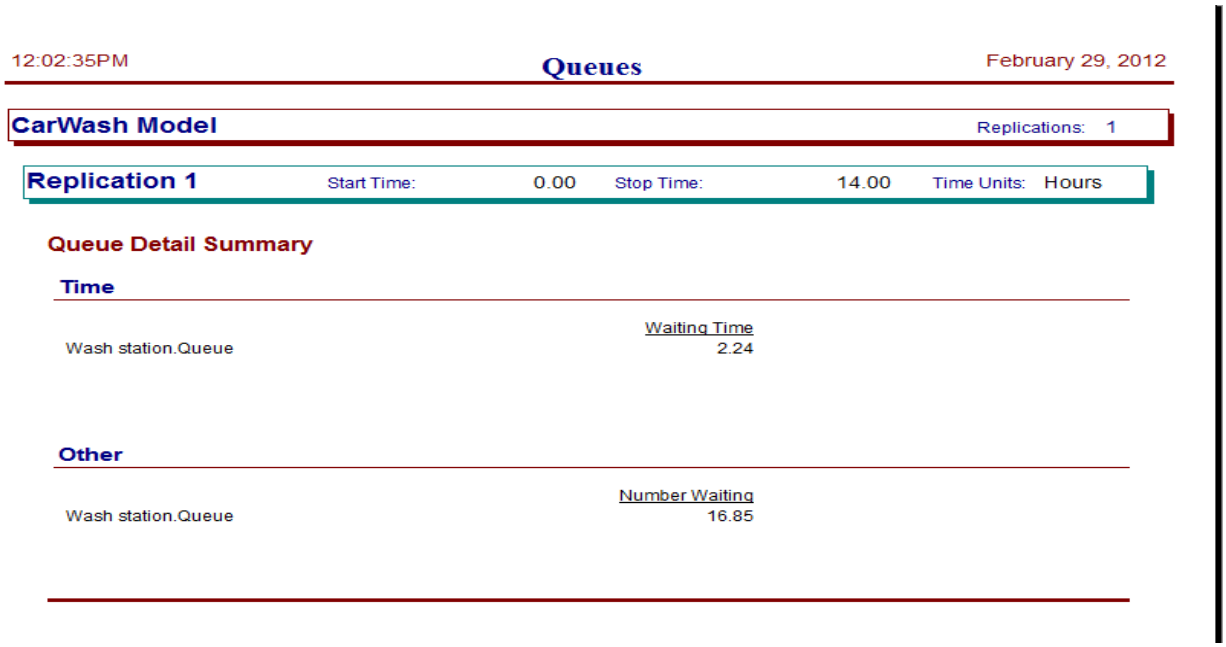
Resource Detail Summary

Usage

	Inst Util	Num Busy	Num Sched	Num Seized	Sched Util
Resource 1	1.00	1.00	1.00	73.00	1.00
Wash	1.00	1.00	1.00	73.00	1.00

Figure 4.19 The Resources report of the Carwash model

Figure 4.20 shows part of the Queues report. The average time that a vehicle spent on the queue was 2.24 minutes. The average number of vehicles waiting in the queue was 16.85.



12:02:35PM Queues February 29, 2012

CarWash Model Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 14.00 Time Units: Hours

Queue Detail Summary

Time

	Waiting Time
Wash station.Queue	2.24

Other

	Number Waiting
Wash station.Queue	16.85

Figure 4.20 The Queues report of the Carwash model

Changing the mean inter-arrival interval to 14.5 minutes, changes the output results. The total number of vehicles serviced becomes 49. The average waiting time will be 1.969 minutes. The resource utilization will be 0.6738 or 67.38%.

The basic animation possible with Arena consists of showing with a small visual object the arrival of an entity and its accumulation in the queue. The default picture for an entity can be changed by editing the Entity data module.

The other animation consists of showing visually the states of the resource, Idle and Busy. To place a picture to indicate each state, the user edits a picture in the Resource Picture Placement dialog by clicking the Resource button in the Animate bar, which is the second tool bar at the top of the Arena window. Arena provides a facility to generate or draw dynamic plots of a simulation run. To define a plot, the user clicks on the Plot button on the Animate bar and enters the data on the Plot dialog window that appears.

Exercises

1. *Write a list of systems that would have a similar model as the one for the Carwash system.*

2. *Increase the inter-arrival interval of the arriving vehicles in the Carwash model and perform one or more simulation models. Comment on the differences in results compared to the results from the original simulation run.*
3. *Decrease the inter-arrival interval of the arriving vehicles in the Carwash model and perform one or more simulation models. Comment on the differences in results compared to the results from the original simulation run.*
4. *Increase the average processing interval of the vehicles in the Carwash model and perform one or more simulation models. Comment on the differences in results compared to the results from the original simulation run.*
5. *Decrease the average processing interval of the vehicles in the Carwash model and perform one or more simulation models. Comment on the differences in results compared to the results from the original simulation run.*
6. *After reading the documentation on how to construct a plot for a simulation run, add a plot to the carwash model. This plot should show the number of vehicles in the system as the simulation time progresses.*
7. *After reading the documentation on how to animation in a simulation run, add animation that shows the state of the wash machine (Idle and Busy). Also include another picture for the vehicles that arrive and wait in the queue.*

4.5 ARENA Data Storage Objects

An important part of the model building process is assignment and storage of data supplied by the user (input parameters) or generated by the model during a simulation run (output observations). To this end, Arena provides three types of data storage objects:

- *Variables,*
- *Expressions, and*
- *Attributes*.

Variables and *expressions* can be introduced and initialized via the *Variable* and Expression *spreadsheet modules*, accessible from the *Basic Process template* and *Advanced Process template*, respectively, in the Project bar.

4.5.1 Variables

Variables are user-defined global data storage objects used to store and modify state information at run initialization or in the course of a run. Such (global) variables are visible everywhere in the model; namely, they can be accessed, examined, and modified from every component of the model. In an Arena program, variables are typically examined in ***Decide modules*** and modified in ***Assign modules***. Unlike user-defined variables, certain predefined Arena (system) variables are read-only (i.e., they may only be examined to decide on a course of action or to collect statistics), and cannot be assigned a new value by the user; the system is solely responsible for changing these values. For instance, the variable ***NQ (Machine_Q)*** stores the current value of the number of entities in the queue called ***Machine_Q***. Similarly, the variable ***NR*** (Machine) stores the number of busy units of the resource called Machine. Other important Arena variables are ***TNOW***, which stores the simulation run's current time (simulation clock), and ***TFIN,*** which stores the simulation completion time.

4.5.2 Expressions

Expressions can be viewed as specialized variables that store the value of an associated formula (expression). They are used as convenient shorthand to compute mathematical expressions that may recur in multiple parts of the model. Whenever an expression name is encountered in the model, it is promptly evaluated at that point in simulation time, and the computed value is substituted for the expression name. Variables of any kind (user defined or system defined) as well as attributes may be used in expressions.

4.5.3 Attributes

Attributes are data storage objects associated with entities. Unlike variables, which are global, attributes are local to entities in the sense that each instance of an entity has its own copy of attributes. For example, a customer's arrival time can be stored in a customer attribute to allow the computation of individual waiting times. When arrivals consist of multiple types of customer, the type of an arrival can also be stored in a customer entity’s attribute to allow separate statistics collection for each customer type.

4.6 ARENA Output Statistics Collection

The end product of a simulation is a set of statistics that estimate performance measures of the system under study. Such statistics can be classified into the two standard categories of ***time averages*** and *customer averages*. More specifically, time averages are obtained by dividing the area under the performance function (e.g., number in the system, periods of busy and idle states, etc.) by the elapsed simulation time. Customer average statistics are averages of customer-related performance values (e.g., customer waiting times in queues).

Arena provides two basic mechanisms for collecting simulation output statistics:

- ***One via the Statistic module, and***
- ***The other via the Record module.***

Time average statistics are collected in Arena via the Statistic module, while customer-average statistics must be collected via a Record module, and (optionally) specified in the Statistic module.

4.6.1 Statistics Collection via the Statistic Module

Detailed statistics collection in Arena is typically specified in the ***Statistic module*** located in the ***Advanced Process template*** panel. Selecting the *Statistic module* opens a dialog box. The modeler can then define statistics as rows of information in the spreadsheet view that lists all user-defined statistics. For each statistic, the modeler specifies a name in the ***Name column***, and selects the type of statistic from a drop-down list in the ***Type column***. The options are as follows:

- ***Time-Persistent*** statistics are simply time average statistics in Arena terminology. Typical *Time-Persistent* statistics are average queue lengths, server utilization, and various probabilities. Any user-defined probability or time average of an expression can be estimated using this option.
- ***Tally*** statistics are customer averages, and have to be specified in a ***Record module*** in order to initiate statistics collection. However, it is advisable to include the definition in the *Statistic module* as well, so that the entire set of statistics can be viewed in the same spreadsheet for modeling convenience.
- ***Counter*** statistics are used to keep track of counts, and like the Tally option, have to be specified in a ***Record module*** in order to initiate statistics collection.
- ***Output*** statistics are obtained by evaluating an expression at the end of a simulation run. Expressions may involve Arena variables such as *DAVG(S)* (time average of the Time-

Persistent statistic S), *TAVG(S)* (the average of Tally statistic S), *TFIN* (simulation completion time), *NR(), NQ(),* or any variable from the Arena Variables Guide.

- ***Frequency*** statistics are used to produce frequency distributions of (random) expressions, such as Arena variables or resource states. This mechanism allows users to estimate steady-state probabilities of events, such as queue occupancy or resource states.

Note that all statistics defined in the *Statistic module* are reported automatically in the User Specified section of the Arena output report. Furthermore, *Queue* and *Resource Time-Persistent* statistics will be automatically computed and need not be defined in the *Statistic module*.

4.6.2 Statistics Collection via the Record Module

The *Record module* is used to collect various statistics. Any statistics related to customer averages or customer observations, such as *Tally* and *Counter*, have to be specified in a *Record module*. Figure 4.21 displays a dialog box for a *Record module*, listing all types of statistics as options in the *Type field*.

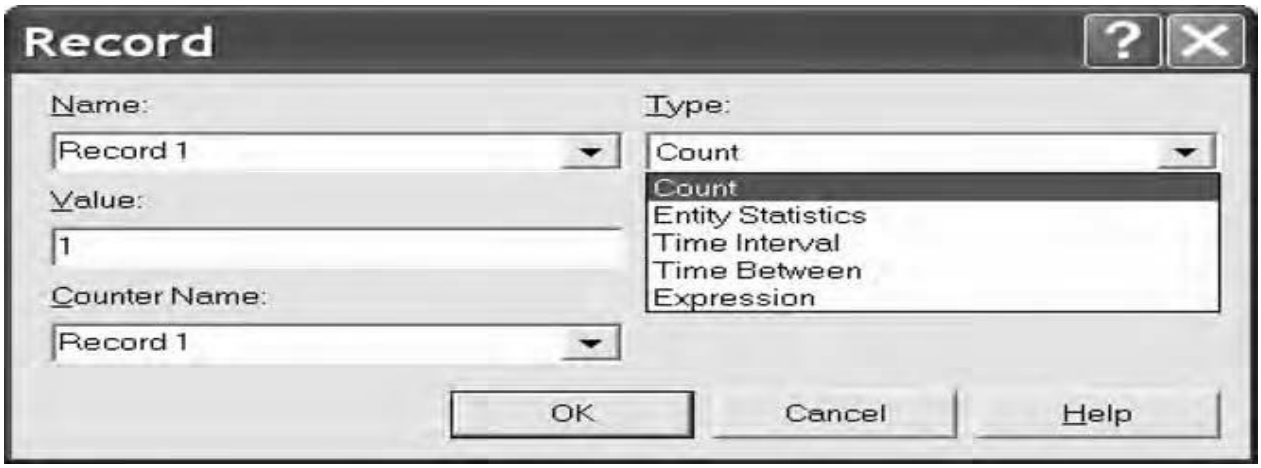


Figure 4.21 Types of statistics collected by the Record module.

These options are as follows:

1. ***The Count*** option maintains a count with a prescribed increment (positive or negative). The increment is quite general: it may be defined as any expression or function and may assume any real value. The corresponding counter is incremented whenever an entity enters the *Record module*.
2. ***The Entity Statistics*** option provides information on entities, such as time and costing/duration information.

3. ***The Time Interval*** option tallies the difference between the current time and the time stored in a prescribed attribute of the entering entity.
4. ***The Time Between*** options tallies the time interval between consecutive entries of entities in the *Record module*. These intervals correspond to inter-departure times from the module, and the reciprocal of the mean inter-departure times is the module's *throughput.*
5. ***The Expression*** option tallies an expression whose value is recomputed whenever an entity enters the *Record module*.

4.7 Arena Simulation and Output Reports

An Arena model run can be initiated and managed in two ways:

- *Via the VCR-like buttons on the Arena Standard toolbar*
- *Via corresponding options in the Run pull-down menu bar*

Standard Arena output reports provide summaries of simulation run statistics, as requested by the modeler implicitly or explicitly.

Arena output reports fall into two categories:

1. ***Automatic reports***: A number of Arena constructs, such as entities, queues, and resources, will automatically generate reports of summary statistics at the end of a simulation run. Those statistics are implicitly specified by the modeler simply by dragging and dropping those modules into an Arena model, and no further action is required of the user.
2. ***User-specified reports***: Reports on additional statistics can be obtained by explicitly specifying statistics collection via the *Statistic module* (Advanced Process template panel) and the *Record module* (Basic Process template panel). Recall that the *Statistic module* is specified in a spreadsheet view, while the Record module must be placed in the appropriate location in the model.

To request a formatted report, the user opens the ***Reports panel*** in the *Project bar* to display a list of report options that correspond to various types of statistics as follows:

- ***Entities reports*** automatically provide various entity counts.
- ***Frequencies reports*** provide time averages of expressions (including probabilities as a special case). The expression must be specified in a Statistic module with the Frequency option selected.

- ***Processes reports*** provide statistics associated with each Process module. These include incoming and outgoing entity counts, average service times, and average delays. Time-oriented statistics (e.g., delays) include the average, half-width of 95% confidence intervals, and minimal and maximal observed values. The half-width value is not displayed if the observations are correlated or if there are insufficient data (in which case Arena prints (insufficient) in the corresponding field).
- ***Queues reports*** provide statistics for each queue in the model, such as average queue delay and average queue size. Additional statistics include the half-width of the 95% confidence interval, and minimal and maximal observed values.
- ***Resources reports*** provide statistics for each resource in the model, such as utilization, average number of busy resource units, and number of times seized.
- ***User Specified reports*** are generated in response to explicit modeler requests for statistics collection in the Statistic module or Record modules. These include any *Time-Persistent*, *Tally*, *Count*, *Frequency*, and *Output statistics*.

Clicking on a report option in the Reports template panel of the Project bar prompts Arena to format the corresponding report and display it in a window.

4.8 EXAMPLE: Two Processes in Series

A two-stage manufacturing model with two processes in series:

Jobs arrive at an assembly workstation with exponentially distributed inter-arrival times of mean 5 hours. We assume that the assembly process has all the raw materials necessary to carry out the operation. The assembly time is uniformly distributed between 2 and 6 hours. After the process is completed, a quality control inspection is performed, and past data reveal that 15% of the jobs fail any given inspection and go back to the assembly operation for rework (jobs may end up going through multiple reworks until they pass inspection). Jobs that pass inspection go to the next stage, which is a painting operation that takes 3 hours per job. We are interested in simulating the system for 100,000 hours to obtain process utilizations, average number of reworks per job, average job waiting times, and average job flow times (elapsed times experienced by job entities). Figure 4.22 depicts the corresponding Arena model and a snapshot of its state.

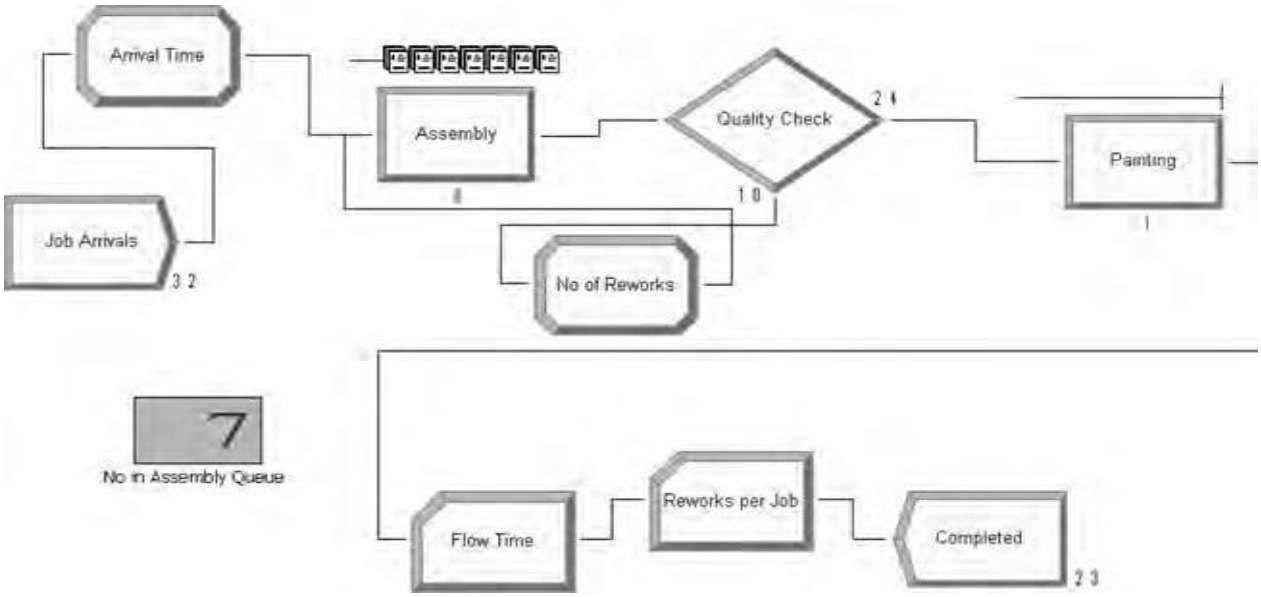


Figure 4.22 Arena model of two processes in series.

In this Arena model, entities represent jobs that are created by the ***Create module Job Arrivals***, and enter the ***Assign module Arrival Time***, where their arrival time is assigned to an attribute called ***ArrTime***. Job entities then proceed to the ***Process module Assembly***, where they undergo assembly. Next, assembled job entities go through inspection in the ***Decide module Quality Check***, and those needing rework are routed to the ***Assign module Number of Reworks*** to record the number of reworks per job, and then back to ***module Assembly***. Job entities that pass inspection proceed to be painted in the ***Process module Painting***, which completes the manufacturing operation. Completed job entities go through ***Record modules Flow Time*** and ***Reworks per Job*** to collect statistics of interest. Finally, job entities enter the ***Dispose module Completed***, where they are removed from the model. The numbers at bottom right of the module icons display the number of entities that departed from the module. The icons above the module ***Assembly*** represent jobs waiting in the Assembly queue, called ***Assembly.Queue***. The Variable window at bottom left of the figure displays the number of the aforementioned jobs. Job icons and data displayed on the module are dynamic, that is, they change as the simulation run unfolds in time. For example, this snapshot indicates that at the time it was taken, 28 jobs were created, 20 jobs were completed, 7 jobs were waiting to be assembled, and 1 job was being assembled.

We next proceed to explain the modules used in the model in more detail (note that the default module names have been replaced here by more expressive names relevant to the problem at hand).

The arrival time assignment is carried out in the ***Assign module Arrival Time***, whose dialog boxes are shown in Figure 4.23.

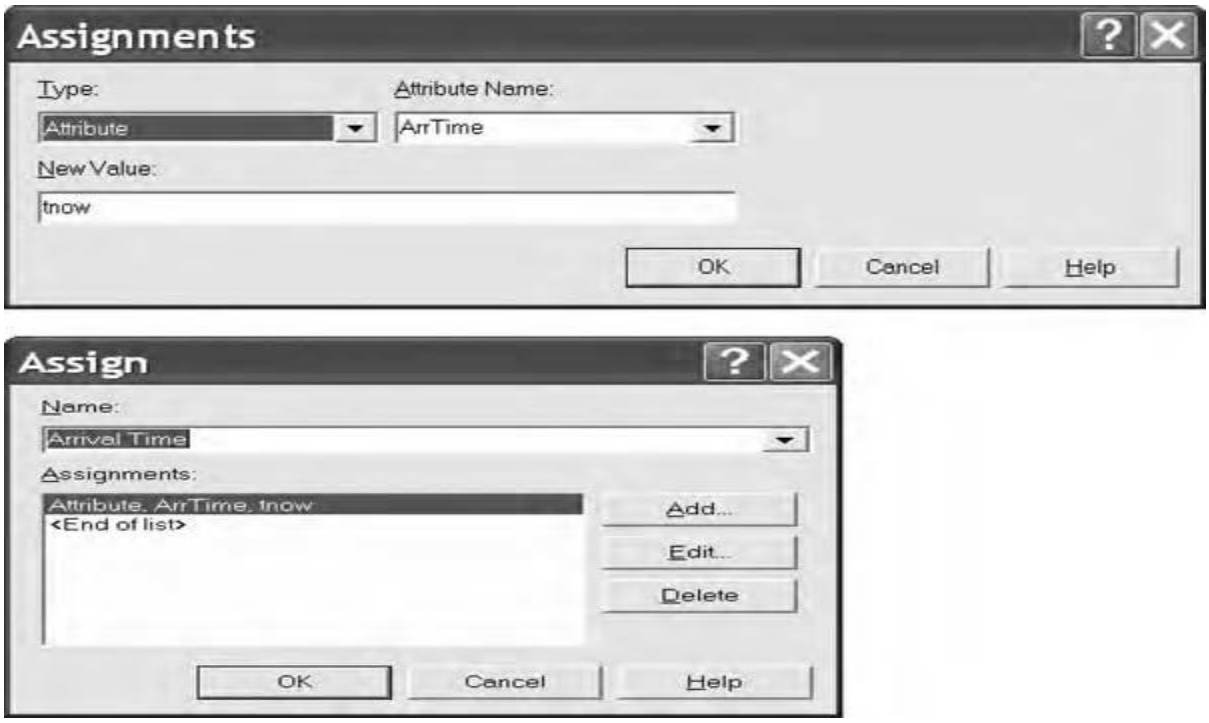


Figure 4.23 Dialog boxes for assigning an arrival time to a job entity's attribute

The bottom dialog box appears first on the screen, while clicking on an item in its Assignments field pops up the top dialog box for entering a value for the corresponding variable or attribute.

The module Assembly in Figure 4.22 is, in fact, a Process module that models the assembly operation with the appropriate job assembly time specification. Recall that the T-bar above the Process icon represents an animated Queue object for holding entities waiting for an opportune condition (e.g., to seize a resource). Note that the T-bar is displayed in a Process module only if the module contains a Seize option. The dialog box of the module Assembly in Figure 4.22 is displayed in Figure 4.23.

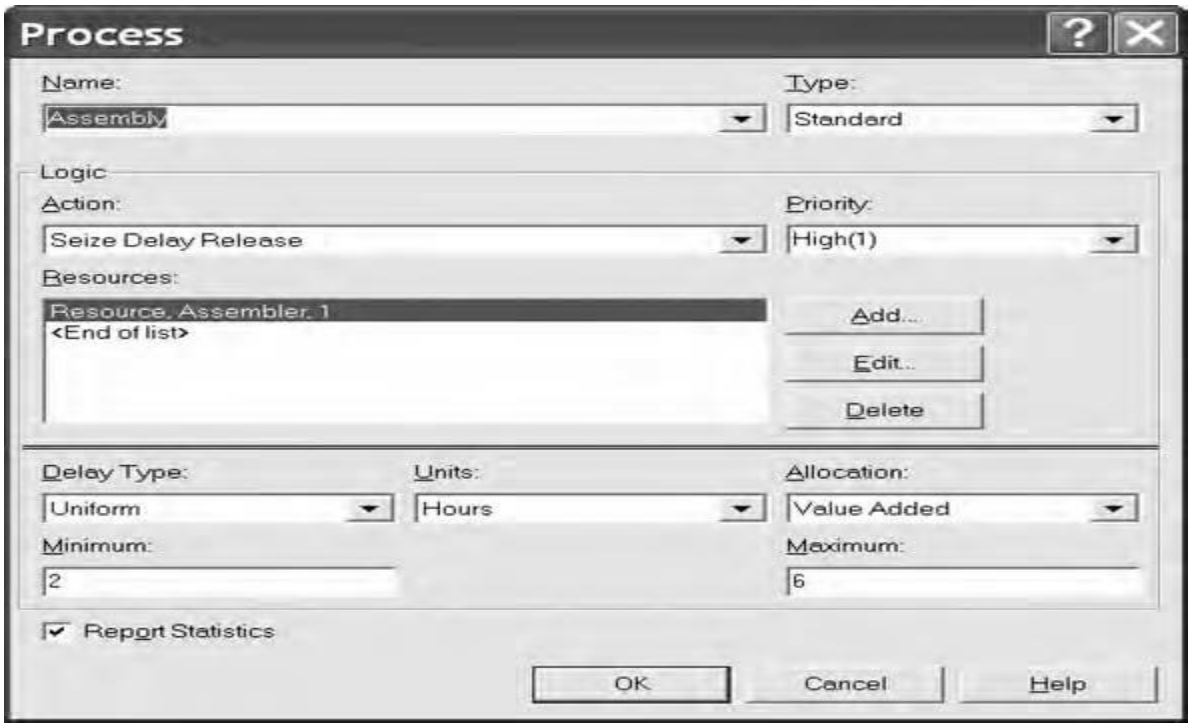


Figure 4.23 Dialog box of the Process module Assembly.

A ***Decide module***, called ***Quality Check***, follows the assembly operation, representing a quality control check. The Decide dialog box is shown in Figure 4.24.

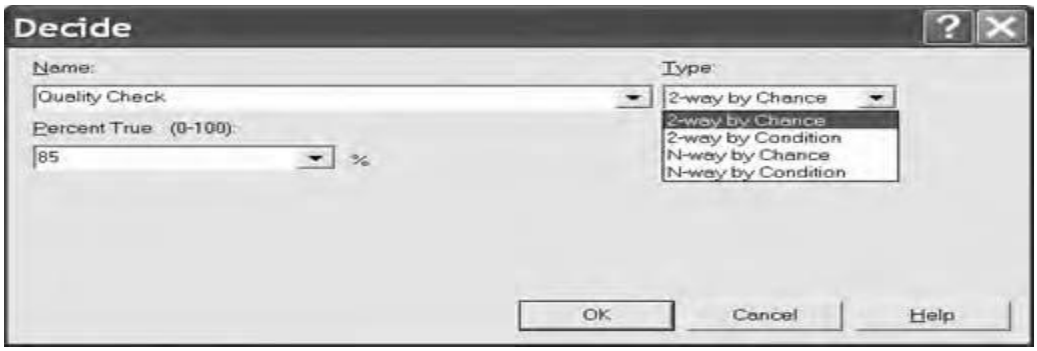


Figure 4.24 Dialog box of the Decide module Quality Check with a two-way probabilistic branching

This module executes entity transfers, which may be probabilistic or based on the truth or falsity of some logical condition. Our example assumes that transfer times between processes are negligible, and therefore instantaneous. Here, we have a two-way probabilistic branching that splits the incoming stream of job entities into two outgoing streams:

- With probability 0.85, an incoming job is deemed "good" (passed inspection), and is forwarded to the Process module, called Painting.

- With probability 0.15 an incoming job is deemed “bad” (failed inspection), and is routed back to the Process module, called Assembly, for rework.

For convenience, the Type field provides separate options for simple two-way branching, which is the most common, as well as general n-way branching, which is more complex. In n-way branching, this module has multiple exits, exactly one of which is designated the else branch and serves for default exits of entities from the module. More specifically, in probabilistic branching, a branch is taken with its associated probability, and the else branch automatically complements the exit probabilities, ensuring that they sum to 1. In conditional branching, the entity exits at the first branch whose condition evaluates to true, and if all conditions are false, the entity exits at the *else* branch.

Job entities that fail inspection enter the Assign module, called No of Reworks, where the job attribute, called Total Reworks, is incremented by the expression Total Reworks + 1. The dialog box for this module is shown in Figure 4.25.

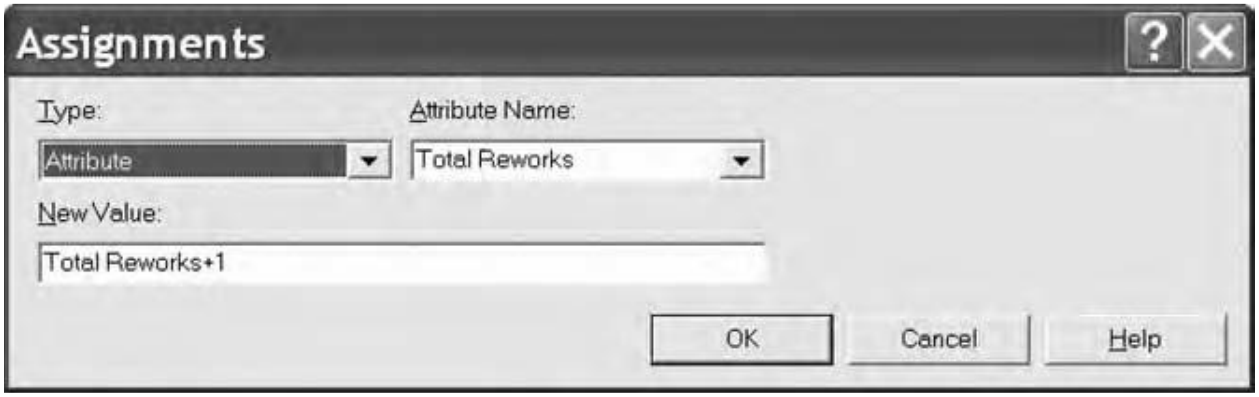


Figure 4.25 Dialog box of the Assign Module No of Reworks.

Job entities departing from the Painting module enter two Record modules called Flow Time and Reworks per Job, to record their flow times and total number of reworks per job, respectively. In our example, the flow time is the difference between the job departure time from the Painting module and the job’s arrival time at module Assembly. Note that the job flow time includes any delays in queues as well as processing times. The average flow time is a customer average (Time Interval statistic), computed internally by a TALLY block, which can be verified by accessing the corresponding .mod file. The dialog box of the Record module is shown in Figure 4.26.

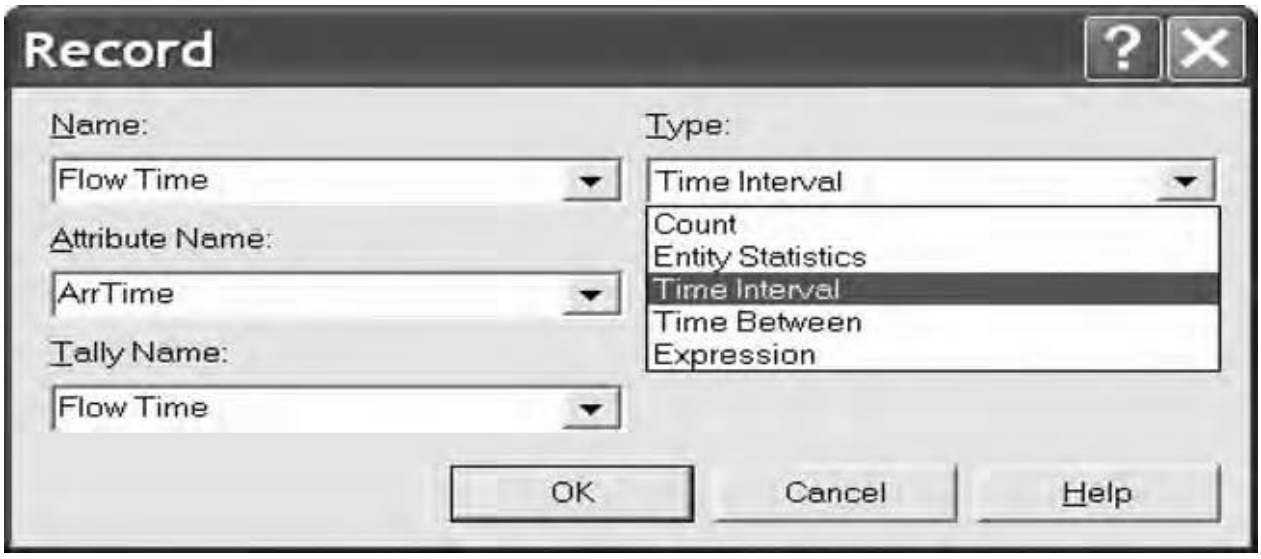


Figure 4.26 Dialog box of a Record module tallying job flow times.

In the next Record module, the expression consisting of the job entity's attribute Total Reworks is tallied in the Record module called Reworks per Job, as shown in the dialog box of Figure 4.27.

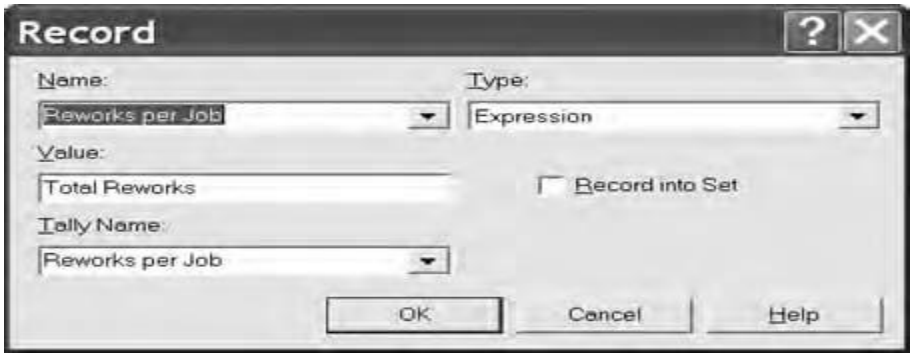


Figure 4.27 Dialog box of the Record module tallying the number of reworks per job.

Finally, job entities proceed to be disposed of in the Dispose module, called Completed. A single replication of the model was run for 100,000 hours, and the results are displayed in Figures 4.28, 4.29, and 4.30.

2:53:05PM **Resources** September 8, 2005

Assembly Op. Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 100,000.00 Time Units: Hours

Resource Detail Summary

Usage

	Inst Util	Num Busy	Num Sched	Num Seized	Sched Util
Assembler	0.94	0.94	1.00	23,417.00	0.94
Painter	0.59	0.59	1.00	19,830.00	0.59

Figure 4.28 Resource statistics for the manufacturing model (two processes in series).

2:53:41PM **Queues** September 8, 2005

Assembly Op. Replications: 1

Replication 1 Start Time: 0.00 Stop Time: 100,000.00 Time Units: Hours

Queue Detail Summary

Time

	Waiting Time
Assembly Queue	35.28
Painting Queue	0.14

Other

	Number Waiting
Assembly Queue	8.27
Painting Queue	0.03

Figure 4.29 Queue statistics for the manufacturing model (two processes in series).

2:54:57PM **User Specified** September 8, 2005

Assembly Op. Replications 1

Replication 1 Start Time 0.00 Stop Time 100,000.00 Time Units Hours

Tally

Expression	Average	Half Width	Minimum	Maximum
Reworks per Job	0.1807	0.006949558	0	4.0000

Interval	Average	Half Width	Minimum	Maximum
Flow Time	49.5252	(Correlated)	5.0004	926.57

Figure 4.30 Flow time statistics for the manufacturing model (two processes in series).

Figure 4.28 shows that the utilization estimates of the Assembler resource at the Assembly module and the Painter resource at the Painting module are 0.94 and 0.59, respectively. These estimates in Figure 4.28 correspond to a heavy traffic regime in the former and a medium traffic regime in the latter. These result in long average buffer delays and large average buffer occupancy in the assembly process, and in short average buffer delays and low average buffer occupancy in the painting process as shown in Figure 4.29. Finally, the Tally section of the User Specified output in Figure 4.30 displays not only averages, but also the corresponding 95% confidence interval half widths, as well as the minimal and maximal observations for the number of reworks and flow time per job. Note that some jobs underwent as many as four reworks, while the average number of reworks is just 0.18, indicating a low level of reworks. The average flow time (49.5252 hours) is moderately longer than the sum of the average delays in the Assembly and Painting queues (35.42 hours) and the sum of average processing times there (7 hours), due to additional rework performed at module Assembly.

4.9 PRACTICAL ASSIGNMENTS

4.9.1 PROBLEM STATEMENT 1

Press operation

The press department of an automobile manufacturing facility runs two main operations, each with its own press machine: front-plate press operation and rear-plate press operation. These operations can be performed in any order, but both have to be performed for each arriving plate. Plates (jobs) arrive randomly and their inter-arrival times are exponentially distributed with mean 5 minutes. The service time in the front-plate press operation is distributed uniformly (1, 5) minutes, and in the rear-plate press operation it is distributed uniformly (2, 6) minutes. A plate joins the queue of the press operation with the least number of plates waiting at that time (since there is no sequencing requirement), and on completion joins the queue of the other press operation after which it departs from the system. Finally, the press department is a three-shift facility running 24 hours a day.

a. Develop an Arena model of the press department, and simulate it for 5 days.

b. Estimate the following statistics:

- Average time arriving plates spend in the press department
- Utilization of the press machine in each operation
- Average queue delay at each operation
- Average time in the press department of those arriving plates that join first the rear-plate press operation, and then proceed to the front-plate press operation

4.9.2 PROBLEM STATEMENT 2

Electrolytic forming process

An expensive custom-built product goes through two stages of operation. The first stage is an electrolytic forming process, served by two independently operating forming machines, where the product is built in a chemical operation that must conform to precise specifications. The second stage is a plating operation in which the product is silver plated. Customer orders arrive with inter-arrival times distributed triangularly (3, 7, and 14) hours, and join a queue in front of the forming process. The electrolytic forming processing time is distributed uniformly (8, 12) hours. The silver-

plating process also has a queue in front of it. Plating time is distributed uniformly (4, 8) hours. The variability in the processing times is due to design variations of the incoming orders. The two processes do not perform perfectly. In fact, 15% of the jobs that emerge from the forming process and 12% of the jobs that emerge from the plating process are defective and have to be reworked. All defective jobs are sent to a single rework facility, where design modifications and corrections are performed manually. However, plating reworks have a lower priority than forming modifications. Plating rework times are distributed uniformly (15, 24) hours, while forming reworks are distributed uniformly (10, 20) hours. Jobs departing from the rework facility go back to the process they came from to redo the operation found defective. Jobs that successfully complete the plating process leave the facility. Note that a job may go back and forth between a process and the rework operation any number of times.

a. Develop an Arena model of the electrolytic forming process, and simulate it for one month (24 hours of continuous operation).

b. How busy are each of the two operations and the rework facility?

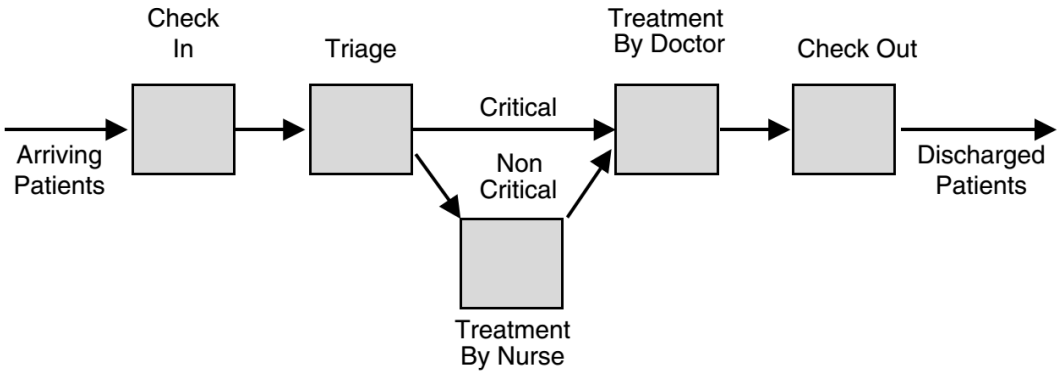
c. What are the expected delays in process queues and the rework facility?

d. What is the expected job flow time throughout the entire facility?

e. Suggest a change in the system to reduce (even slightly) the expected job flow time. Run the modified model and compare the job flow statistics

4.9.3 PROBLEM STATEMENT 3

The emergency room of a small hospital operates around the clock. It is staffed by three receptionists at the reception office, and two doctors on the premises, assisted by two nurses. Figure below depicts a diagram of patient sojourn in the emergency room system, from arrival to discharge.



Patients arrive at the emergency room according to a Poisson process with mean inter-arrival time of 10 minutes. An incoming patient is first checked into the emergency room by a receptionist at the reception office. Check-in time is uniform between 6 and 12 minutes.

Since critically ill patients get treatment priority over noncritical ones, each patient first undergoes triage in the sense that a doctor determines the criticality level of the incoming patient in FIFO order. The triage time distribution is triangular with a minimum of 3 minutes, a maximum of 15 minutes, and a most likely value of 5 minutes. It has been observed that 40% of incoming patients arrive in critical condition, and such patients proceed directly to an adjacent treatment room, where they wait FIFO to be treated by a doctor. The treatment time of critical patients is uniform between20 and 30 minutes.

In contrast, patients deemed noncritical first wait to be called by a nurse who walks them to a treatment room some distance away. The time spent to reach the treatment room is uniform between 1 and 3 minutes and the treatment time by a nurse is uniform between 3 and 10 minutes. Once treated by a nurse, a noncritical patient waits FIFO for a doctor to approve the treatment, which takes a uniform time between 5 to 10 minutes. Recall that the queueing discipline of all patients awaiting doctor treatment is FIFO within their priority classes, that is, all patients wait FIFO for an available doctor, but critical patients are given priority over noncritical ones. Following treatment by a doctor, all patients are checked out FIFO at the reception office, which takes a uniform time between 10 and 20 minutes, following which the patients leave the emergency room.

The performance metrics of interest in this problem are as follows:

- Utilization of the emergency room staff by type (doctors, nurses, and receptionists)
- Average waiting time of incoming patients for triage
- Average patient sojourn time in the emergency room
- Average daily throughput (patients treated per day) of the emergency room

To estimate the requisite statistics, the hospital emergency room was simulated for a period of 1 year.