AN ABSTRACT OF THE THESIS OF

Guangchao Zhang for the degree of Master of Science in Industrial and Manufacturing Engineering presented on March 15, 1993.

Title: *An Object-oriented Simulation System for Softwood Sawmills*

Abstract approved: Redacted for Privacy

Sabah U. Randhawa

S3 (Softwood Sawmill Simulator) is a sawmill simulation system for modeling the operations of Pacific Northwest softwood lumber mills. S3 consists of three main parts. The first part is the framework for construction of the sawmill layout. The second part focuses on individual machine centers, their process and down times, and their interconnections. The third part consists of databases for raw material and final products. S3 inputs process logic from external data files. All parts are integrated in an object-oriented framework. The system was developed using the object-oriented environment, Actor. All data input and output are through database files in dBASE IV format. S3 can model a sawmill represented by the machine center and connection types defined in S3. The size of the model is controlled by the Actor programming environment. The construction of a sawmill model is demonstrated.

An Object-oriented Simulation System for Softwood Sawmills

by

Guangchao Zhang

A THESIS

submitted to

Oregon State University

In partial fulfillment of
the requirements for the
degree of

Master of Science

Completed March 15, 1993

Commencement June 1993

APPROVED:

Redacted for Privacy

Major professor, Industrial and Manufacturing Engineering

Redacted for Privacy

Department head, Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented March 15, 1993

Typed by Guangchao Zhang for Guangchao Zhang

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

An Object-oriented Simulation System for Softwood Sawmills

I. INTRODUCTION

This thesis describes S3 (Softwood Sawmill Simulator), a personal computer based simulation system developed to model the operations of softwood sawmills.

I.1. Problem Context

Sawmills, like other manufacturing enterprises, are value-added industries; profit is the primary motivation for operations. Sawmills in North America have been experiencing increasing costs both in raw material and operations. To extract maximum value from the raw material and to process the material in the most efficient way are the primary concerns of sawmill management.

Sawmill operation is the process of log breakdown into lumber and material of smaller dimensions. Log breakdown and log throughput are two major factors affecting the sawmill performance. The effect of log breakdown is straightforward; the higher the lumber recovery, the higher the revenue. The log throughput manifests itself through machine utilization and material flow in sawmills. Higher throughput can significantly affect the profitability of a mill.

The desire to improve sawmill performance by optimization of the above two factors has been the subject of many studies on sawmill systems. This is evidenced by the large number of computer simulation applications for the sawmill process. However, much more attention has been paid to the lumber recovery issue than to the dynamics of sawmill operations. The reasons include: (1) the effect of log breakdown on yields is more apparent than that of mill dynamics; and (2) modeling of log breakdown involves large amounts of computations but a smaller scope of technical undertaking, while modeling of sawmill dynamics requires the knowledge not only the sawmill operations, but also the analysis tools such as simulation and statistical methods.

The optimization of log breakdown and log throughput may become competing goals in many circumstances. Therefore, a sawmill simulation system must deal with both factors together in order to strike a balance that offers the best overall results.

I.2. Research objective

The ultimate goal of S3 is to provide the sawmill management with a sawmill simulation system to analyze the effects of log breakdown and log throughput. During the design of new mills or modification of existing mill layout, analysis of material flow can help identify potential bottlenecks, balance the workload, and generate the highest possible throughput. In daily mill operations, the effect of different log process logic on output may be simulated before application, thus helping the management select the process that gives the best recovery and throughput.

A number of steps must be taken to accomplish this goal (Figure 1). These represent a transition from the programmer at the functionality level to the user at the usability level.

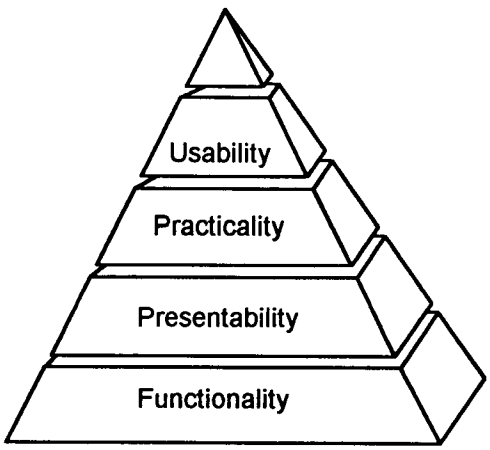


Figure 1. Design goals of S3

Functionality — This is the ability to construct a model from common sawmill components, and to execute the simulation. A simulation framework is needed to perform this function, by driving data input through a model and analyzing the material flow. A set of building blocks are needed to provide the components that contain the necessary data constructs and methods to observe system behaviors. The framework and building blocks together supply the functionality of S3.

Presentability — This is the ability for S3 to present information to the users. A rudimentary presentation may be simple text files. More sophisticated presentations may include tables, charts, graphics, audio/video effects, and virtual reality.

Practicality — This is the ability for S3 to respond to the user input and changes in the system in a reasonable time frame and with desired accuracy and precision.

Usability — This is the ability of S3 to accept inputs from users in a manner natural to the human user, to simulate the realistic complexities in sawmill operations, and to present outputs in a similar manner. This requires intelligence in S3 to interpret, reason, and recover from errors in user input, and to "talk the language" of sawmill management in presentation of outputs. This also requires S3 to model any characteristics in sawmill operation that the users wish to investigate.

In this research, the focus in developing S3 is functionality. This step is critical for the other higher level functions to be attained. This system developed also provides enough presentability to facilitate the development and testing of the program.

I.3. Thesis organization

This thesis consists of five chapters. In the introduction chapter, the problem and the scope of the project are outlined. The second chapter provides background in sawmill operation, existing research in the area of simulation as applied to sawmill operations, and a description of what is considered a complete model of a sawmill. The third chapter is dedicated to the implementation of S3, its overall structure and building blocks. The fourth chapter walks the reader through the building of a model in S3 and execution of the model. In the conclusion chapter, S3 is evaluated in terms of model implementation, limitations, and further enhancements.

II. BACKGROUND

This chapter covers background in sawmill operations and in simulation, reviews the existing research in modeling sawmill operations using discrete event simulation, and specifies the requirements for S3.

II.1. Sawmill operation

Sawmills differ from most other manufacturing processes because there is no assembly involved; there is only the reduction of the raw material into smaller dimensions. The raw material possesses a large degree of variation that consequently appears in the final product. Special process logic is applied that accounts for variability of raw material in order to increase the value of final products. The variability in process logic as a result of variability in raw material in turn affects the material flow in the sawmill. Deciding the degree to include these, and other, variability is understandably the most complex aspect of designing a sawmill model.

Sawmill operations can be characterized by their raw material and final product types, type of machinery, or the log process logic that dictates how raw material is processed into final products. Williston [1990] provides a general reference on sawmill operations. The type of material a sawmill processes affects the type and layout of the machinery it uses.

Sawmills in North America fall into three major categories. Softwood dimension lumber sawmills (small log mills) process small logs as raw material, feature high throughput machinery and simple log process logic, and produce limited types of end products. Softwood grade lumber sawmills (large log mills) use larger logs, have machinery designed for cutting high grade lumber using complex process logic, and produce a variety of end products. Hardwood sawmills (grade lumber mills) generally produce grade lumber and are similar to the softwood grade mill.

Logs are the basic raw material of a sawmill. Primary attributes include size (both cross section and length), shape, grade and species. The sawmilling process mechanically converts those logs into lumber and by-products.

Lumber prices and customer orders are two factors that help determine the final product mix, with several species often mixed in one product category. The log process logic can include a weighting scheme devised to determine the desirability of a particular final product type, based

on market price and demand. In sawmills, lumber is usually sorted by size into groups of the same cross section size and length. Grade, size, and species determine lumber prices. Residues from the conversion process include bark from debarking, sawdust from sawing, chips from slabs, edgings, and trimmings, and shavings from planing.

A typical sawmill derived from Williston [1990] includes the following process components:

Log yard — The log yard provides the storage and infeed of raw material to the production lines. Crane or forklift trucks move logs from the log yard to the log decks of process stations.

Log bucking — Some of the long logs need to be sawn into shorter lengths. The reasons for bucking include: providing process stations with material of suitable lengths; removing defects from the logs; or optimizing the yield of raw material. Not all sawmills require log bucking.

Head sawing — At this stage, logs are sawn into smaller pieces such as cants and flitches that are further processed into lumber, or into pieces of final dimension directly. Head saw stations can have single, double, or quadruple saw bands that can make multiple cuts in one pass. A double-bladed band also permits sawing in both the forward and return passes. Single band saws offer more versatility in log processing logic, while multiple band saws provide greater throughput. Depending on the process logic, multiple passes may be needed to completely process a log.

Resawing — The resawing process complements the head sawing by performing processes that are not economical or cumbersome for head saw stations to perform. A resaw station also receives material that needs to be resawn. Both single and double band resaw stations are available.

Cant edging — Cant edging is the process of sawing a cant into multiple boards in a single pass. A cant edger usually features multiple saw blades, adjustable gaps between blades, and one pass process.

Board edging — Board edging is the process of ripping a piece length wise to remove rough edges, or to make boards of narrower widths. Most of the pieces with rough edges require an edging process. Board edging is performed at a board edger or a combination edger capable of both cant edging and board edging.

Trimming — During the trimming stage, pieces are cut across length to remove rough ends or defects, or cut into one or more pieces of lumber with desired lengths. Almost all pieces generated in the sawmill process require trimming. Trimming is done at trimmer stations.

Sorting — The sorting stage is where lumber is sorted by species, grade, length, width, thickness, or cross section. There are usually a number of manual or automatic sorter stations in a sawmill.

The process times at each machine are stochastic in nature and related to the characteristics of the machine and the attributes of the material processed. Three types of sawmills are considered with regard to their differences in process speed.

Some older sawmills, originally designed to process large logs, find themselves in the midst of declining diameters in raw material supply. The equipment in these mills is overpowered for smaller logs. To simulate this type of sawmills, the process time may not have much correlation with material attributes.

Process times vary the most in large log grade sawing mills. Head saws in these mills are designed to adjust feed speed with the height of saw kerfs. The process time, therefore, may be proportional to the area of face sawn.

Modern high speed and high throughput sawmills are designed for small logs and equipped with high power machines. The process speeds in these machines are nearly constant. The process time is merely proportional to the length of material. This is the type of sawmill most often found in softwood dimension sawmills.

Process times are subject to operator's performance and the need to balance material flow in sawmill. When there is a backlog of material waiting to be processed, the operator tends to speed up the operation. When blockage occurs at the upstream machine center, or no material is available at the down stream machine centers, the pressure is on the current machine center to accelerate the process, even change the original process pattern, to smooth the flow of material.

Machine process time is stochastic. It was observed that process times often follow a log normal distribution [Adams, 1987]. It is possible to record empirical data to formulate a log normal distribution to represent the process time.

The desire to optimize on lumber recovery is often compromised by the need for greater log throughput. The best log breakdown strategy may not fill a customer's order. The process logic may be changed to accommodate change in material flow. Both the raw material cost and the operation cost contribute to the costs of sawmill process.

II.2. Simulation

Simulation is the process of driving inputs (often random) through the system and observing the system's dynamic behavior. Simulation is an experimental tool that derives sample data and statistical estimates of the system being modeled, thus reducing the need for direct experimentation on the real system or for the development of analytical solutions.

Simulation has several advantages over analytical solutions or direct experimentation. Deriving a feasible solution with analytical models usually requires greater simplifying assumptions. A simulation model can better represent the real world because fewer restrictive assumptions are required. In addition, simulation can provide solutions when analytical models fail to do so, for example, solutions to complex queuing problems in sawmills. Actual experimentation generally costs more than simulation; simulation may thus be the only practical approach, as in modeling log process logic. Although simulation is not an optimization tool, it provides a method for exhaustively exploring many possible solutions when no simple analytical search for the optimum is available.

The disadvantages associated with simulation generally refer to the precision and accuracy of the results and to the cost of creating the model. Because simulation uses sampling procedures to estimate system parameters, its outputs are only statistical estimates. However, if the distributions used in simulation are realistic representations of the actual system parameters, the estimation can come close to the true value of the system; otherwise, it is inaccurate.

In most cases, the cost of using simulation will be greater than an analytical solution, if available, because simulation models are usually more complex. The increased complexity is due to the increased interactions and interdependencies that can be represented in a simulation model.

II.2. Literature review

A variety of operations research methods have been applied to the analysis of log process logic, and material flow [Holmes, 1976]. Among them, discrete event simulation is the most common

method for analyzing the dynamics of a sawmill's production flow. A number of researchers have focused on simulation modeling of material flow in sawmills, thus providing sawmill management with a method of analyzing changes before they are implemented.

A simulation system can be either one of two classes: simulation languages or manufacturing simulators [Law and Kelton, 1991]. Simulation languages offer unlimited modeling flexibility, but require programming expertise and long coding times. A simulator provides quick model development capabilities but is limited to the features it provides.

In the literature review, existing research on sawmill simulation will be reviewed to cover these aspects: 1) programming environment; 2) user environment; 3) material flow control; and 4) sawmill layout design.

A general-purpose sawmill simulation system is designed to model a variety of sawmills differing in size, raw material supply, and production pattern. It generally contains a raw material database, generic log process logic, and flexible sawmill layout design.

Aune [1973, 1974] reported a simulation modeling package (MILLSIM) with an interactive design program that was used in the analysis of a planned sawmill. On the basis of the model results, the mill design was modified to eliminate several material-flow problems.

DESIM (DEsign SIMulator) by Adams [1984a, 1984b, 1985, 1988] was implemented in the GASP IV simulation language [Pritsker, 1974] for simulating hardwood sawmills. The users provide raw material data, machine data, and log sawing data. Using this information, DESIM simulates the sawmilling operation and generates a summary of raw material usage and final product production, as well as the statistics for each machine center's performance. The user may change the input parameters and rerun the program to compare modifications to the sawmill's operations.

SPSM (Southern Pine Sawmill Model) was developed by Wagner and Taylor [1983] using SLAM II simulation language [Pritsker, 1986]. A more recent version, MICRO-MSUSP [Wagner, Seale and Taylor, 1989], implemented the modeling package to microcomputers, added a graphical interface, and incorporated the "Best Opening Face" (BOF) [Hallock and Lewis, 1971] program for log breakdown. The model covers log bucking, debarking, head sawing, resawing, edging, and trimming based on look-up tables generated from the BOF program. An expert system is built in as a check on the sawmill layout and to indicate when the user makes an illogical connection in the layout.

Hall and Jewett [1988] demonstrated a minicomputer-based sawmill simulator using the SLAM II simulation language and TESS (a graphical post-processor of SLAM II) that graphically displays sawmill layout and model output.

Kline and Weidenbeck [1989] developed a personal computer based simulation package for hardwood sawmills featuring graphical animation. It was based on SIMAN simulation language [Pegden, 1989] and CINEMA (a graphical post-processor for SIMAN).

Meimban [1991] reported a SIMAN based simulation model that includes log conversion logic based on log quality, process flow, and process and down time. The model reports the capability to differentiate and qualify the volume of sawmill products.

Many other simulation models focus on a particular sawmill or type of sawmill. They are generally the manufacturing simulator type, "hardwired" to particular patterns of sawmill layout, raw material breakdown, and production. They include models that: test the design of live-sawing hardwood sawmills [Martin, 1971]; model the effects of varying log supply characteristics on sawmill productivity [Aune, 1973]; analyze the material flow among the machine centers and buffers in a sawmill [Aune, 1974]; design a new sawmill and identify optimum log size distributions and material flow [Briggs, 1978]; evaluate a sequential production system in the design of a rough mill for furniture interior parts [Araman, 1977] ; simulate a softwood small-log sawmill [Kempthorne, 1978] and a hardwood dimension sawmill [Penick, 1969]; model the effects of controlled log sequencing on piece flow and volume production in a pine sawmill [Richard, 1974]; and redesign a multi-pass head saw sawmill [Orbay, 1984].

Almost all the manufacturing simulator type of models claim some degree of general applicability. However, they generally do not allow users to easily and readily modify the layout or process logic.

II.3. System requirements

For modeling purposes, a sawmill can be considered as a number of linked process stations. A process station generally has one or more surge decks for temporary storage of incoming material, a sawing unit that processes the material, and a process logic that determines how materials are processed. The process logic also determines the type of output and process time. It may be provided as a machine operator's real time decision or as preset look-up tables.

The types of process stations used in a particular sawmill are determined by the sawmill type and log process logic. In the sawmilling process, logs are first debarked at debarker stations and cut to length at bucking stations or vice-versa. Head saws perform the primary log breakdown, while edger, resaw and trimmer stations perform the secondary breakdown. Final products are sorted and packed at sorter stations. Chipper stations are used to reduce slabs into chips.

Aside from process stations, there are two types of stations for handling the multiple input/output connections between process stations. The collector station handles the multiple input / single output links. The transfer station handles the single input / multiple output links. The function of a transfer is to route incoming material to different destinations according to the attributes of the material. Sometimes the routing criteria include considerations such as balancing the in-process inventory at various machine centers. For example, when a process station is down, the preceding transfer will route materials to other work stations to prevent stoppage of the material flow.

All stations are linked to one another by connections. The connections include conveyors (continuous) and transports (intermittent). Conveyors installed in a sawmill are generally in the form of belt or roller conveyors. Transports often include forklifts or cranes.

In the design of a sawmill simulation model, it is always desirable to make the model more representative of the real system. However, there are limits as to what level of realism a model can have. First, the current level of knowledge and technology is an important factor in determining how realistic a sawmill simulation model can be. Second, an increase in model realism always results in an increase in model complexity and since the complexity of a model determines the levels of resources that are required to develop the model, model realism is limited by the resources available.

In the case of sawmill simulation modeling, the following areas are identified as most essential to a realistic sawmill model: raw material distribution, log process logic, machine process time, and material flow.

The data for raw material include log species, grade, and shape information. Log species affects process variables such as shrinkage and final product value. Log shape information is essential for determining how breakdown will occur. The level of realism in representing log shape can vary greatly, with the simplest form being a cylinder described by diameter and length. The next step in realism is a truncated cone which requires an additional parameter, taper. The realism of a log shape can be further increased with the addition of sweep, eccentric cross section, or even

the real shape of the log. Presently, a log is usually modeled as a truncated cone, which holds enough realism yet is easy to implement. The log grade is determined by both internal and external defects, which are very difficult to characterize and quantify, and has received less attention until recently.

A machine center's process time has many stochastic elements, some of which are determined by the type of material being processed. Relating the process time to the material attributes increases the realism of a model, but it also greatly increases the simulation's computation time. It can be advantageous to model process time using probability distributions rather than material attributes.

The piece flow through a sawmill is enormous. In a sawmill model, it may not be necessary to model the flow of each and every piece. Therefore, it may be possible to eliminate material flow to make the model run faster.

The observations from a sawmill simulation fall into three categories, material tally, resource utility, and throughput. Statistics from the observations of materials can be used to derive log and lumber tallies by species, grade, and sizes. Material tally also includes revenues and costs of production. Statistics on resource utility include utilization of every work station and surge deck, and process times at each work station. Throughput statistics include material flow through each work station by piece count, linear footage, or volume.

A complete sawmill model tries to integrate all the basic components of a sawmill system. The combined effect of material flow and process logic is not visible unless they are modeled together. The process logic partially determines the flow of material in a sawmill. On the other hand, process logic can be adjusted to smooth out the flow. The objective of sawmill operations is to extract the maximum value from the raw material, while keeping the operational cost at a minimum. Only a model that combines these two aspects will provide the most effective model tool.

S3 only simulates the operations from the head saw log deck to the sorter. Since S3 relies on external data input for process logic, and the data set presently available to S3 does not include debarking and bucking, these two operations are not included in the current development of S3.

III. SYSTEM STRUCTURE

This chapter, grouped into four sections, presents a detailed treatment of S3. The first section discusses the programming and operating environment, together with the benefits and shortcomings of object-oriented programming as applied to the development of S3. The second section illustrates the framework of S3 and the internal structure of the system. The discussion of the building blocks for S3 is in the third section. The fourth section outlines the requirement for external data files required to run the system.

III.1. Operating environment

III.1.A. Operating platform

S3 runs under Microsoft Windows (TM) version 3.1 [Microsoft, 1991], a graphical operating environment for personal computers. S3 was developed in Actor (TM) version 4.0 [Whitewater, 1991a], a Microsoft Windows application that provides an object-oriented programming environment.

A design goal for a sawmill simulator is to provide sawmill management with a useful tool to investigate sawmill design and operations. The choice of personal computers as the hardware platform offers several advantages. Personal computers are inexpensive, widely available, and have the largest user base. Many sawmills use personal computers for office work and production monitoring and control. Sawmill management is generally familiar with personal computers.

Microsoft Windows is a graphical extension to Microsoft DOS (TM) [Microsoft, 1990], the disk operation system that most personal computers are using. Newer versions of Microsoft Windows will be able to run on a number of hardware platforms. This cross platform capability can be taken as granted by many Microsoft Windows Applications, including S3. The choice of Microsoft Windows as the software platform for S3 leads to the advantages of a graphical user interface, universal peripheral support, and access to a large amount of computer memory.

III.1.B. Object-oriented programming environment

Object-oriented programming techniques share many similarities with discrete event simulation. In discrete event simulation, entities receive and act upon events. In an object-oriented environment, objects receive and act upon messages. Law and Kelton [1991] provide a thorough treatment of simulation. Cox [1986] gives a complete coverage of object-oriented programming.

Actor is a pure object-oriented development system hosted on Microsoft Windows. It features an object-oriented language with syntax similar to that of Smalltalk [Goldberg and Robson, 1989], one of the pioneering pure object-oriented languages. Actor is an interpretative environment that generates stand alone applications. The current version of Actor includes database support through an interface with the Q+E database library [Whitewater, 1991b]. S3 uses Actor's database facilities for data input and output. The choice of Actor as the development environment for S3 offers the advantages of object-oriented programming and database facilities, in addition to the benefits of the Microsoft Windows operating platform.

Developed in an object-oriented environment, S3 is a collection of Actor classes interacting with one another. These classes can be functionally grouped into interface, framework, and building blocks (node, entity, event classes) (Figure 2). Each class has an interface to the external database engine for loading and saving data.

Interface — The S3 application and S3 window classes define the interface. Most of the data input/output are through database files.

Framework — This group of classes define the framework for a discrete event simulation. Their main functions are to drive inputs through the system, and record statistical observations. These classes are discussed in detail in the section on "Framework".

Building blocks — This group of classes include the node classes, the entity classes, and the event classes. They form the building blocks used in developing representations of sawmill systems. They are discussed in detail in the section on "Building blocks".

The four primary concepts used in an object-oriented programming environment are: data abstraction, inheritance, polymorphism, and encapsulation.

Abstraction This is a process of concentrating on features that help understand a system, while ignoring ones that are irrelevant. The process of specifying classes and function groups in S3 is a process of abstraction.

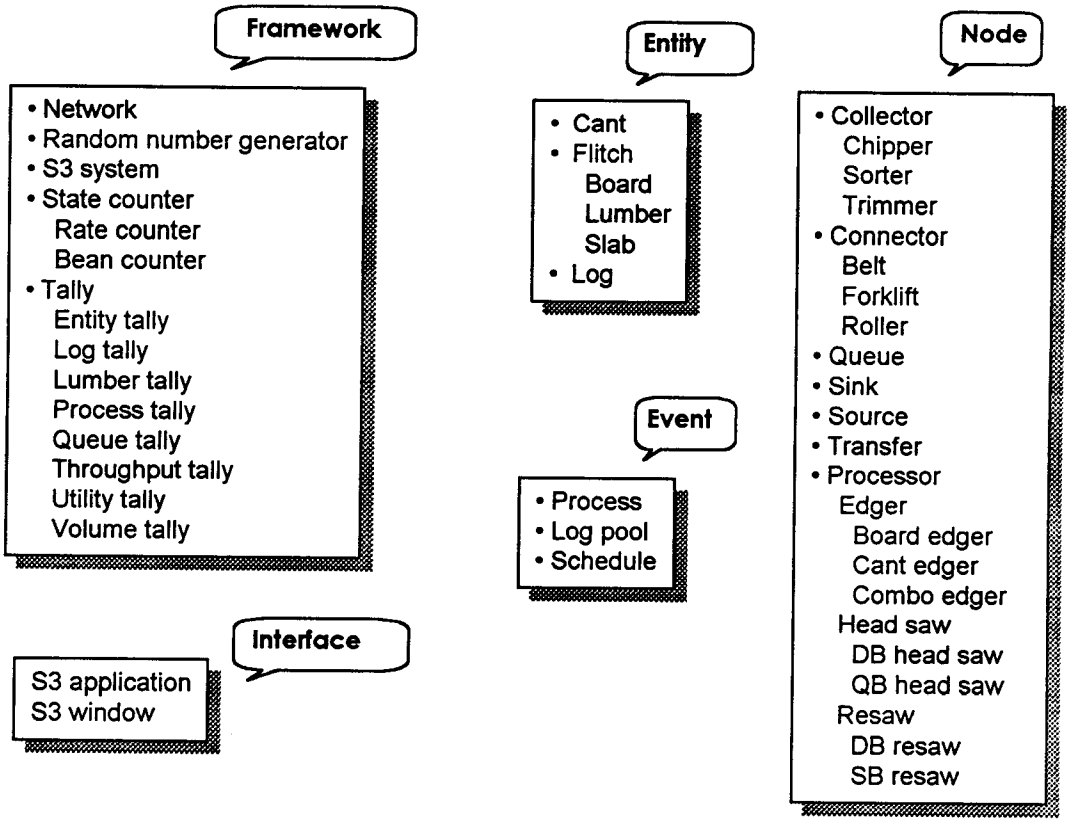


Figure 2. The class hierarchy of S3

Encapsulation The primitive element of object-oriented programming is an object. An object contains information about that entity it represents and the operations in which the entity participates. This is to say that an object encapsulates both data and functions. In S3 most data items are localized to their own class. Those data items requiring global visibility are localized to the class "S3 System".

Inheritance This refers to the ability of one class to define the behavior and data structure of its instances as a superset of the definition of another class or classes. In other words, a subclass in a hierarchy inherits properties from its superclass; in addition, the subclass may add properties specific to itself. This is illustrated in the class hierarchy for the processor node class in Figure 2. The processor class defines data and functions dealing with process times that all subclasses inherit. Each subclass (edger, head saw, etc.) then redefines these operations (such as

"beginSaw" and "endSaw" methods) to add specific requirements for the node type.

Polymorphism This refers to the ability of two or more classes to respond to the same message, each in its own way. The sender need not be concerned with methods that are executed as a result of the message that is sent. For example, each node class needs to respond to the message "process" when an entity arrives at the node, and each node processes the entity in ways specific to the function of the class. A queue node either sends the entity to the down stream processor node, or places it in the queue. A processor node looks into the process attached to the entity and starts the operation sequence specified in the process. A transfer node looks at the type of the entity and the process attached to the entity to determine the output node where the entity is to be routed.

III.2. Framework

The framework of S3 consists of classes for performing discrete event simulation, including the network, the system clock, the random number generator, the message post, global variables, and generation and recording of statistical observations. These classes provide the shell of a discrete event simulator. Together with the building blocks discussed in later sections, The framework lays the foundation for the functionality of S3.

III.2.A. Network

The network class defines the topology of the sawmill layout. It contains a source node, a sink node, and a container for all nodes loaded from the data file. Individual nodes contain information on their input and output nodes. All nodes can be traversed starting from either the source node or the sink node. Network performs the operations of addition, deletion, and connection of nodes.

To input a sawmill model from database files, S3 first loads in the network database file. It then reads the names of individual nodes in the network and creates them one by one. After S3 creates a node, it loads from a database file all data pertaining to that node. After all nodes are created and node data loaded, S3 reads the network database again to obtain data about the connections among nodes and creates these connections. Data for the connections are loaded

as each connection is created. Log distributions are loaded into the source node. At this stage S3 has loaded all the necessary data for the sawmill model to run.

To generate reports for the simulation run, S3 asks the network to save statistics to the report database file. The network in turn signals all the nodes to save their statistics to the report file. Basically, time stamped snapshots are made for all nodes in the network. One of the advantages of an object-oriented environment, message passing, is put to good use in the data handling. The network just sends a message to a node to save statistical observations. The individual nodes generate node specific statistics .

III.2.B. System clock

The system clock consists of three clocks: the previous time, the current time, and the next time. They provide references to the timing of the last event, the current event, and the next event. The system clock has a precision of one second.

At the start of simulation, all three clocks reset themselves to time zero. During the simulation, they advance to the times of their corresponding events. The next time is used as a look-ahead for events. For example, when a process event is going to take place, it must post an "end of process" future event. The next time is set to the time of this future event. S3 can use this next time to see if any down time is scheduled within the time frame. If there is a down time that is going to interrupt the scheduled process, S3 can raise a flag at the specified station for interrupt handling.

S3 does not use the notion of stop time in its system clock. Once S3 enters the autonomous execution state, it will continue removing and dispatching messages in the message post (discussed in section on "Message Post"), until the message post becomes empty, or an event sets S3 to the sleep state.

III.2.C. Random numbers

Random number generation is an essential component in the simulation of dynamic systems. In order to generate a random sample from a given random distribution, a random number generator usually generates a uniform random number between the values of 0 and 1, then transforms this number into a random sample from the distribution.

S3 uses Actor's built in random number generator to obtain a uniform random number, then uses distribution transform functions to transform this number into a random sample from a statistical distribution. S3 defines transform functions for the following statistical distributions:

Uniform — A uniform random distribution is specified by its minimum and maximum. S3 uses the shift and scale method to transform a unit random number to a sample from a uniform distribution. One application for the uniform distribution is the start time of a machine breakdown, since it is totally unpredictable.

Exponential — S3 uses the inverse transform method [Knuth, 1981] to generate a sample from an exponential distribution, specified by the mean of the distribution. Exponential distributions are generally used in determining the time intervals between random arrivals of entities. It is possible to use this distribution to generate process times.

Normal — A normal distribution is specified by its mean and standard deviation. S3 uses the Box-Muller method [Knuth, 1981] to generate a sample from normal distribution. Normal distribution can be applied to a wide variety of circumstances such as processing times, maintenance times, or down times.

Log normal — A log normal distribution is specified the same way as a normal distribution. To generate a random sample, a sample is taken from a normal distribution and its logarithm is taken as the output. It has been observed that the process times in process stations follow a log normal distribution [Adams, 1987].

Line distribution — A line distribution is an empirical numerical distribution specified by a collection of numerical pairs. Each pair has a numerical value and a probability of occurrence. A random sample from a line distribution may be an interpolation of the numerical values specified. Line distributions are used in the generation of log attributes that are numerical, such as diameter and length.

Step distribution — A step distribution is an empirical symbolic distribution specified by a collection of symbol / numeric pairs. Each pair has a symbolic value and its associated probability. A random sample from a step distribution must be one of the symbols specified in the parameters. Step distributions are used in the generation of log attributes that are symbolic, such as log grade and species.

In S3, the process times for process stations may be generated from theoretical distributions such as a normal or an exponential distribution. Since process time cannot be negative, S3 uses the rejection method to discard negative samples, until a non-negative value is generated from the distribution.

The log parameter distributions, on the other hand, are based on empirical distributions. S3 uses line distributions for numerical parameters such as log diameter, and step distributions for symbolic parameters such as log species.

III.2.D. Message post

Two types of message passing mechanisms are used in S3. One is the message passing directly between nodes. The other is postponed message passing via the message post.

The message post is a message collection sorted by the time stamp of the messages. Each message is posted by a node with a time stamp that tells when to dispatch itself, a destination node, and an entity to send to the destination node, or an action to be enacted at the destination node.

The message post uses a looping mechanism to dispatch messages. This dispatching process continues as long as all of the following conditions are true: 1) the system is in an awake state; 2) the system is not in the stepping mode; 3) the message post is not empty.

When a message is about to be dispatched, the current system clock advances to the time stamp of that message, and the message is sent to the destination node. Then the next message is ready to be dispatched. Table 1 lists a trace segment of the messages associated with the processing of the first log at the start of a simulation run. For clarity of demonstration, nodes are connected to each other without any connectors, with the exception of a connector that links the source node and the transfer node called "Log-kicker". Each message line in Table 1 contains the following items:

Now — This is the time when the message activates. It is expressed in hour:minute:second format. The system clock in S3 has a granularity of one second.

Message — This identifies the message. S3 uses three types of messages. The first type is an event that changes the system states. These messages are identified by their

lack of sender nodes and future times and are executed immediately. The second type is the scheduled messages to be invoked at future times. These messages are posted by their sender nodes to be performed on the recipient nodes. The third type of message dispatches entities from one node to another. The entities use a combination of entity type and entity tag as identification.

Sender node This is the name of the node that dispatches an entity or posts a message.

Recipient node This is the destination node for the dispatched entities and the scheduled messages.

Future This is the time a scheduled message is delivered to the recipient node.

In the following sections, the messages in Table 1 are explained in chronological order.

Time 0:00:00 This is the initial schedule put into the message post before the simulation starts. The message "beginRun" is posted to the source node to be invoked at time zero. The message "endRun" is invoked at time 8:00:00; it signals the end of simulation.

When the source node receives the "beginRun" message, it generates a log entity, Log0000, and then posts a "log" message scheduled at time 0:00:09. The log entity is sent to the transfer node, log kicker, which routes it to the conveyor leading to the quadruple band head saw. The log entity is scheduled to reach the end of the conveyor at time 0:00:11.

Time 0:00:09 The "log" message reaches the source node. As a response, the source node generates a log entity, Log0001, and then posts a "log" message to itself, scheduled to arrive at time 0:00:18. The log entity, Log0001, goes to a transfer node, Log kicker, where it is routed to the conveyor leading to the double band head saw. The log entity starts the travel on conveyor and is scheduled at finish the travel by time 0:00:19.

Time 0:00:11 The "endTravel" message reaches the conveyor LK-QBHR-roller. The conveyor sends Log0000 to the buffer deck of the head saw. Since the buffer deck is empty and the head saw is idle, the buffer deck sends the log to the quadruple head saw which starts loading the log entity. The end of the loading is scheduled at time 0:00:13.

Table 1. Examples of message passing

#	Now	Message	Sender node	Recipient node	Future
1	0:00:00	beginRun	Source	Source	0:00:00
2	0:00:00	endRun	Source	Source	1:00:00
3	0:00:00	beginRun		Source	
4	0:00:00	log	Source	Source	0:00:09
5	0:00:00	send Log0000	Source	Log-kicker	
6	0:00:00	send Log0000	Log-kicker	LK-to-QBHR-roller	
7	0:00:00	beginTravel		LK-to-QBHR-roller	
8	0:00:00	endTravel	LK-to-QBHR-roller	LK-to-QBHR-roller	0:00:11
9	0:00:09	log		Source	
10	0:00:09	log	Source	Source	0:00:18
11	0:00:09	send Log0001	Source	Log-kicker	
12	0:00:09	send Log0001	Log-kicker	LK-to-DBHR-roller	
13	0:00:09	beginTravel		LK-to-DBHR-roller	
14	0:00:09	endTravel	LK-to-DBHR-roller	LK-to-DBHR-roller	0:00:19
15	0:00:11	endTravel		LK-to-QBHR-roller	
16	0:00:11	send Log0000	LK-to-QBHR-roller	QBHR-deck	
17	0:00:11	send Log0000	QBHR-deck	QB-headsaw	
18	0:00:11	beginLoad		QB-headsaw	
19	0:00:11	endLoad	QB-headsaw	QB-headsaw	0:00:13
20	0:00:13	endLoad		QB-headsaw	
21	0:00:13	beginPosition		QB-headsaw	
22	0:00:13	endPosition	QB-headsaw	QB-headsaw	0:00:15
23	0:00:15	endPosition		QB-headsaw	
24	0:00:15	beginSaw		QB-headsaw	
25	0:00:15	endSaw	QB-headsaw	QB-headsaw	0:00:16
26	0:00:16	endSaw		QB-headsaw	
27	0:00:16	send Flitch0004	QB-headsaw	QBHR-drop-sorter	
28	0:00:16	send Flitch0004	QBHR-drop-sorter	Board-edger-collector	
29	0:00:16	send Flitch0004	Board-edger-collector	Board-edger-deck	
30	0:00:16	send Flitch0004	Board-edger-deck	Board-edger	
31	0:00:16	beginLoad		Board-edger	
32	0:00:16	endLoad	Board-edger	Board-edger	0:00:18
33	0:00:16	send Flitch0005	QB-headsaw	QBHR-drop-sorter	
34	0:00:16	send Flitch0005	QBHR-drop-sorter	Board-edger-collector	
35	0:00:16	send Flitch0005	Board-edger-collector	Board-edger-deck	
36	0:00:16	send Slab0002	QB-headsaw	QBHR-drop-sorter	
37	0:00:16	send Slab0002	QBHR-drop-sorter	Chipper	
38	0:00:16	send Slab0002	Chipper	Sink	
39	0:00:16	send Slab0003	QB-headsaw	QBHR-drop-sorter	
40	0:00:16	send Slab0003	QBHR-drop-sorter	Chipper	
41	0:00:16	send Slab0003	Chipper	Sink	
42	0:00:16	send Log0000	QB-headsaw	Sink	
43	0:00:16	beginUnload		QB-headsaw	
44	0:00:16	endUnload	QB-headsaw	QB-headsaw	0:00:19

Time 0:00:13 The "endLoad" message is activated at the QB-head saw. The QB-head saw starts the position operation for Log0000, and schedules the end of the position operation at time 0:00:15.

Time 0:00:15: The "endPosition" message is activated at QB-head saw. The QB-head saw starts the sawing operation for Log0000, and schedules the end of operation at time 0:00:16.

Time 0:00:16 The "endSaw" message is activated. Four new entities are generated as a result of the quadruple band sawing. Two flitches go through the QBHR drop sorter to the board edger collector, then move to the buffer deck for the board edger. Since the board edger is idle and the buffer is empty, the first flitch, Flitch0004, goes to the board edger right away. The board edger starts the loading operation and schedules the end of loading at time 0:00:24. The second flitch, Flitch0005, stays in the buffer deck.

The two slabs, Slab0002 and Slab0003, are destined for the chipper. They move through the drop sorter for the head saw and reach the chipper node. The chipper node tallies them and then sends them to the sink node.

In the mean time, the log entity Log0000 no longer exists. It goes to the sink node to be tallied. The QB-head saw now starts the unloading operation to unload the cant left in the node, and schedules the end of unloading operations at time 0:00:19.

III.2.E. Global variables

All global variables are defined as class variables of the class S3 system. They include:

Main window The main display window is used to display the progress of data storage and retrieval, and on-going messages.

Trace file When the users instruct S3 to save trace text to a trace file, the information displayed in the main window is also saved into the trace file.

System clock Three variables, the previous time, the current time, and the next time, are available in the system.

System state The system state can be asleep or awake.

Message post The sorted collection for messages posted by nodes. It also has a looping code block that continuously looks for next message and dispatches it.

Network This variable points to the network of nodes created from the database files. Individual nodes are accessed through network.

The SQL This variable is the liaison to the database extension in Actor. S3 passes the appropriate data to the SQL, which in turn handles the physical storage and retrieval of databases.

Process pool When a process logic is needed the first time, it is loaded from a database file and kept in the process pool. The next time it is called, this cache can provide the process log without reloading from the database file.

III.2.F. Statistical observations

S3 records and generates reports on three types of system statistics: material tally, resource utilization, and system throughput. Nodes in the system employ statistical observation units, tallies, to record system statistics. Tallies, in turn, rely on more basic units, counters, to collect statistical observations.

Counter

Counters are the basic observation units for collecting statistical observations on system states or entity properties. A counter generates descriptive statistics about an observation. It gives the minimum, mean, and maximum values observed, as well as the standard deviation and the number of observations. S3 uses three types of counters to account for different types of statistical observations.

State counter A state counter collects statistics on observations. When a value is passed to a state counter, the counter compares the value with the minimum and maximum values recorded, and if necessary, updates these values. It then updates the sum and the sum of square recorded for all observations, and increments the number of occurrences. The sum and sum of square values, together with the

number of occurrences, are used to derive the mean and standard deviation of the observations.

Rate counter — A rate counter uses the moving average principle to record the rate of occurrence at the time of an observation. A rate counter contains a moving band that keeps several observations in chronological order. When the number of observations reaches a preset limit, the band is ready to provide the rate data. When a new observation is added to the band, the oldest one in the band is discarded, and the time intervals between consecutive observations are averaged. The observed values, such as piece volumes, are also averaged. The ratio of the two, the rate measurement, is recorded as a statistical observation into a state counter. When requested, a rate counter provides statistics on the rate the observations are occurring.

Bean counter — A bean counter only remembers the number of occurrences of an observation. S3 uses the Bag class in Actor to implement the bean counter. A bag object is capable of keeping multiple occurrences of the same elements. A bean counter only generates number of occurrences for an observation as statistics.

Tally

Tallies are the statistical observation units employed in a node to record various statistics for that node. A node can have one or more tallies to record throughput, utilization, and its process states.

Entity tally — An entity tally uses a bean counter to record the counts of different type of entities. Each machine center has an entity tally to record the piece count of entities passing through the node.

Log tally — A log tally includes bean counters for species, grade, diameter, length and taper. The result of a bean counter is a histogram. In order to produce a histogram for numerical values, such as diameter, the value is rounded off to an integer. Only the source node has a log tally.

Lumber tally — A lumber tally includes bean counters for species, grade, width, thickness, cross section, and length. As in log tally, numerical values are rounded to integers in

order to generate histograms. The sorter node, where lumber is sorted, contains a lumber tally.

Process tally A process tally includes state counters for load, position, saw, unload, and return operations. A process station has at least the saw operation defined. The other operations are not always applicable to all types of process stations. Only process stations have process tallies.

Queue tally A queue tally includes state counters for wait time, queue size, queue states (empty, full, overflow, in use). Only a queue node can have a queue tally.

Throughput tally A throughput tally includes rate counters for piece, linear, and volume throughputs. A piece throughput only accounts for the number of pieces observed in a given time period. A linear throughput looks only at the length-wise measure of the observations. A volume throughput records the volume of pieces observed.

Utility tally A utility tally includes state counter for busy, idle, and down states for a machine center. Whenever a node is engaged in some activity, it is said to be busy. To be idle means the node is ready but no activity is available to it. A down state signifies the node is not ready for activities.

Volume tally A volume tally includes a state counter for the volume of a piece. It is primarily used to observe the flow of by products, such as slabs. In S3, the chipper station has a volume tally to record the volume of slabs.

III.3. Building blocks

The building blocks of S3 complement the framework by providing the basic elements to build a simulation model. The network uses nodes to construct a model for a sawmill. During simulation, events provide the driving force to run the model, and process logic dictates the generation of new entities and their flow among the nodes. The system clock and message post govern the event execution. The random number generator supplies event timing. The statistical observation units in individual nodes record system states and entity flow. Together, framework and building blocks construct the model of sawmill specified in database files, and observe the behavior of the model in a simulation run. The sections below examine the building blocks of S3: entities, process logic, nodes, and events.

III.3.A. Entity

An entity can be of type log, cant, flitch, slab, board, or lumber. The meaning of each type is slightly different from the common usage. Each of the types will be discussed in the context of entity. Whenever appropriate, the common usage is quoted in parentheses. Three dimensional variables describe the entity dimensions. Currently, these dimensions are recorded in nominal units.

These six entity types can sufficiently describe all the entities encountered in a common sawmill. When drying and surfacing are introduced into the model, new attributes may be added to an entity to represent dried and surfaced characteristics of lumber. Chips, bark, trimmings, shavings, and sawdust are not explicitly represented in S3. As a result, the volume statistics on end products usually do not match that of the logs.

Log — Log is the raw material to the sawmill. A log is first processed into cants and slabs. Sometimes, flitches are also generated from a log. A log entity has a dimension of diameter, length, taper.

Cant — When a log has one or more faces opened, it becomes a cant ("flitch", "timber"). A cant can have 1, 2, 3, or 4 opened faces. Usually a cant is further processed into boards and slabs at a cant edger. A cant has a dimension of width, height, length.

Slab — A slab is a piece of wood cut from a log as a result of opening face ("slab"), edged off a flitch from an edging operation ("edging"), or trimmed from a board from a trimming operation ("trimming"). It is usually sent to a chipper station. Sometimes, especially when a slab is cut from a log, a flitch ("jacket board") may be sawn from a slab. Dimensions of a slab are width, thickness, length.

Flitch — A flitch is a piece of wood cut to the final thickness but needs edging and trimming. One or more boards may be cut from a flitch through the edging operation, depending on the process logic. Dimensions of a flitch are width, thickness, length.

Board — A board is a piece of wood cut to the final thickness and width, only needed to be trimmed to final length. Dimensions of a board are width, thickness, length.

Lumber — When a board is trimmed, it becomes a piece of lumber. Lumber is the final product of sawmilling process. A piece of lumber ("board", "timber") has all the final dimensions set. Further process such as drying and planing may be performed on a piece of lumber. Dimensions of a lumber entity are width, thickness, length.

The life span of an entity varies according to its type, time of creation, and process type. With the exception of logs, all other types of entities are the result of process logic applied to logs. When a process logic is performed, one or more new entities are created in S3. For example, a log is processed into two slabs and a cant. After the process, the old entity (the log, in this example) no longer exists; therefore, it moves to the sink node directly to record statistics and then leaves the system. Each entity is time stamped at creation, thus its age can be traced at any point in time.

III.3.B. Process logic

The process logic plays a crucial role in the simulation of a sawmill. Sawmilling is essentially the breakdown of raw material into smaller and numerous pieces. This breakdown is dictated by the process logic. The design of process logic to recover more values from the raw material has been the subject of many studies. S3 uses process logic as user input; no analyses or observations are applied to it. However, S3 can provide feedback on the result of a set of process logic as they are applied to the dynamic operations of the model.

A process logic is a sequence of operations defined for entities of particular dimensions. The first operation in the sequence applies to logs. Each subsequent operation applies to the entities resulting from the preceding operation. An operation defines its two consequences. The first contains the type, quantity, and dimensions of new entities to be generated, and the operation for each new entity. The second pertains to the remaining portion of the old entity. All parameters specified for a new entity also apply to the old entity. The operation sequences come to an end when resulting entities require no further operations, as in the case of a lumber or slab entity.

A process logic applies only to logs of certain dimensions and quality. Presently, the process logic available to S3 is the BOF sawing logic [Hallock and Lewis, 1971] that considers log diameter, length, and taper. In the process database, each process logic has a unique process tag. Within a process logic, each operation sequence carries a sequence tag and defines the

new and old entity dimensions and the sequence tags of corresponding operations for the entities.

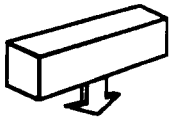
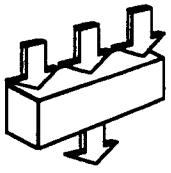
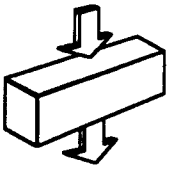
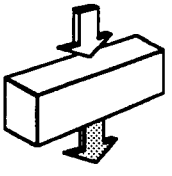
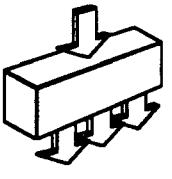
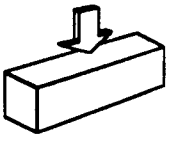
When a log is generated, it immediately looks for its process logic. The process pool in the S3 system provides a dictionary of process logic keyed to log diameters. The log looks up the dictionary to find the identification of the process logic. If the process logic is already loaded into the dictionary, it is returned to the log; otherwise, the process logic is retrieved from the database file and returned.

III.3.C. Node

The node class is a generalization of all stations and connections. A node has a name, a description, and the names of input and output nodes that link to this node. The node class keeps statistics on material throughput, resource utilization, and provides a message passing mechanism for inter-node communications.

The node class is an abstract class; specialized nodes descend from it to inherit its general behavior and add specialized behavior. Table 2 presents the major node classes.

Table 2. Node classes for S3

Source	Source node is the starting node of a network. It feeds new entities into the network. The log generator is in this node.	
Collector	Collectors specialize in multiple input connections. There is no limit on number of inputs a collector node can take.	
Queue	Queues simulate the behaviors of buffer decks for process nodes.	
Processor	Process nodes contain methods to execute process logic and process time delays. For example, sawyers and edgers simulate the sawing (head sawing and resawing) and edging (cant and board edging) operations.	
Transfer	Transfer nodes specialize in multiple outputs. Current limitation on number of outputs is four. Built-in routing criterion is used for deciding the passage of entities to different outputs.	
Sink	Sink node is the ending node for the network. It receives all the entities destined to leave the system, tallies them and disposes of them properly.	
Connector	Connectors represent the physical connections between machine centers. Generally, nodes inside a machine center do not use connectors.	

A "machine center" in S3 is represented as a group of nodes (Figure 3): a collector to handle inputs, a queue node to handle the buffer deck, a process node to handle the sawing process, and a transfer node to handle the output. Nodes within a "machine center" connect directly with each other. Connectors are used to link "machine centers".

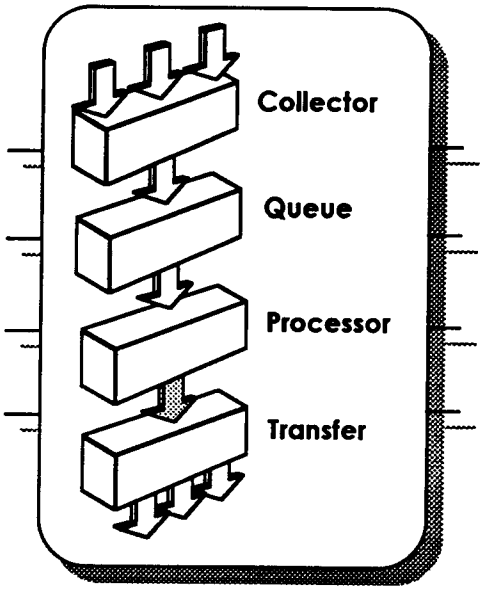


Figure 3. A typical "machine center" in S3

The following sections examine these node types in greater detail.

Collector

Collectors serve a unique purpose of supporting multiple inputs. They are used whenever multiple input connections are called for. There is no preset limit on the number of inputs. Collector nodes usually serve as the front end of a "machine center".

Chipper — A chipper node only receives slab entities. It records the volume of slabs. The chipper class inherits the behavior of multiple inputs from the collector class. In sawmills, drop sorters transfer slabs to a master conveyor leading to the chipping station. For simplicity, the slab outlet in a transfer node can connect to a chipper node directly.

Sorter — A sorter node only receives lumber entities. It contains a lumber tally to record statistics on lumber entities. It usually has a trimmer node as input. However, transfer nodes can send lumber entities directly to sorter, hence the reason for multiple inputs in a sorter node.

Trimmer — A trimmer node only receives board entities that need to be trimmed. Transfer nodes often provide an outlet for board entities to go directly to a trimmer node. This requires the trimmer node to handle multiple inputs.

Connector

Connectors link "machine centers" in a sawmill model. A connector has an input and an output. The travel of entities is simulated as time delays determined by travel distance and speed. A connector may be a belt conveyor, roller conveyor, or forklift.

Continuous movement of entities is readily simulated by a connector. Intermittent movement (batch load / unload) can be simulated by a connector preceded by a queue node. Upon the arrival of an entity, the queue node checks queue size. When the number of entities in the queue reaches a preset limit, the queue node sends all entities it holds to the connector node. These entities will reach the output node of the connector simultaneously.

Queue

Queue nodes simulate the behavior of buffer decks in sawmills. Queue nodes have preset capacities on number of entities (fixed or flexible), and use the FIFO discipline (first in first out) ranking criterion. Using the entity counts instead of the physical sizes of entities is a simplification in S3. Entities, especially flitches and boards overlap in the deck. This factor is hard to model, and is not included in S3. The implication of this simplification is that the capacity of queue is larger than actually needed.

Upon the arrival of an entity, the queue node dispatches the entity to its output node (usually a process node) if it is idle; otherwise, the queue node retains the entity. When the output node becomes idle, it sends a message to the queue node that it is ready to receive an entity.

A queue node can be empty (no entities in queue), in use (entities in queue), full (size of queue equal to fixed capacity), or overflow (size of queue over flexible capacity). A full queue does not receive new entities, while an overflowed queue can continue accepting new entities. A queue tally records statistics on these states, waiting time of entities in queue, and the size of the queue.

Edger

The nodes that represent the edging operations generally feature one pass operations, are preceded by a queue node, and followed by a transfer station.

Board edger — A board edger node only receives flitch entities. It has an operation sequence of "load - saw - unload". The results of the process are the generation of one or more boards and slabs.

Cant edger — A cant edger node only receives cant entities. Similar to board edger, it defines an operation sequence of "load - saw - unload". The process generates boards, flitches and slabs.

Combo edger — A combo edger node can receive either cant or flitch entities and processes them as if it is a cant edger or a board edger.

Sawyer

Sawyer nodes represent sawing stations. They require a position operation after loading an entity, and permit a return operation on entities.

Head saw — Head saw can have two (twin band) or four (quad band) saw blades. A process logic may specifically ask for twin band or quad band. The operation sequence is usually "load - position - saw - return - unload". A head saw node only accepts log entities.

Resaw — Similar to the head saw, a resaw can have one (single band) or two saw blades. The operation sequence is usually "load - position - saw - return - unload". A resaw can accept any type of entity requiring processing. It is the most versatile process node type of all.

Transfer

Transfer stations are nodes that simulate the behavior of drop sorters in sawmills. Transfer nodes have up to four output connections, in order to mimic the manual drop sorters used in sawmills. More connections can be added by modification of this class. Upon arrival of an entity, the transfer node takes a number of steps to determine which of the four destinations to dispatch the entity. First, it checks if there is still a process attached to the entity. If one is attached, it extracts the type of process node specified, then locates the output that leads to such a process node. Second, if no process is to be applied to the entity, the transfer checks the type of the entity (slab, board, or lumber) and dispatches the entity to its designated nodes (chipper,

trimmer, or sorter). Failure to locate an output usually indicates a logic error in the network layout (the sawmill layout in the model) and causes the transfer node to signal an error because it cannot dispatch the entity as programmed. Other more sophisticated methods for dispatching an entity can be added to assign destinations not specified in the process. This is useful in situations of machine breakdown where the flow of material must be redirected and the process logic recalculated.

Source

The source node serves as a counterpart for log storage in a sawmill. Only one source node exists in a network. A source node contains a log distribution that generates logs at time intervals specified as constant or samples from a statistical distribution, to provide the initial and subsequent entity arrivals to the network. When a log is generated, a process logic is immediately attached to it. The S3 system class has a process logic pool for holding process logic read from a process database file. Specifics of assigning a process logic to a log are discussed in the section on "Process logic".

Sink

The sink node does not have a physical counterpart in sawmills. It serves as the termination node of a network and receives all entities flowing through the network. Therefore, it is a good place to collect the statistics about the overall system output. Only one sink node can exist in a network, and it does not have any output connections. However, it can have as many input connections as required. Because of this property, sink class is inherited from the collector class. Data collection in sink class is done by an entity tally that collects data on entities processed by the network.

III.3.D. Event

Events are the driving force in a simulation model. During execution, events generate new entities and new events to perpetuate the simulation. Log generation is an event to generate log entities as input to the system. Schedule events control the changes in system states. Process events control the generation of new entities and their movement in the system. These events are discussed in detail in the following sections.

Log generation

Log generation is handled by the log pool inside the source node. Two events take place when the source node makes a log entity. First, in response to the "log" message, a random log sample is created from the log distribution, which is then sent to the output of the source node, be it a connector or a queue. Second, a new "log" message, designated for the source node, is posted to the message post, to be enacted at some future time.

Two conditions may arise that call for different strategies in log generation. When the main concern of the simulation is to study the material flow for a given time frame, it is desirable to simulate an endless log supply. In this case, the log distribution database provides user defined distributions for log species, grade, diameter, length, and taper. To generate a log, a random sample is made from each of the distributions and the random samples are combined together to make a log.

The second condition arises when the primary focus is to simulate the processing of the log to identify a better process logic for a given group of logs. A log pool is constructed by prefabricating a log entity for each real log in the study, with all the required attributes. In this simulation, the source node in response to the "log" message, will pick a log entity from log pool in a predetermined manner. The simulation will end when all the log entities in the pool are processed. Additional runs can be made with the same log pool. The manner in which a log entity is selected from the log pool can be totally random, or follow a specified procedure. For example, logs can be sorted by their diameters and selected in ascending, descending, or other specified order, or logs can be selected in groups of like attributes.

Schedule

Schedules are events that change system states at specific points of time. A schedule has a name, the name of the target node, a start time, a duration, and a flag indicating if the event is preemptive. All schedules defined by the users are loaded into the S3 system, in the form of messages in the message post, before the simulation starts. If no target node is specified, the event goes to the source node. If a schedule is preemptive, it executes the event immediately, whether or not the target node is engaged in another event. A non-preemptive schedule event can adjust its event time so that it is executed right after the current event in the target node. Schedules usually come in pairs, one to begin a new system state, the other to end it.

To run the model, the user must define at least two schedules. The first is "start run", with the source node of the network as the target. It starts at time zero, and has no duration. This is the trigger event to start the simulation. When the source node receives this event, it starts generating the first log and schedules the next log arrival. The second required schedule is "end run", also targeted for the source node. The "end run" schedule breaks the message looping process in the message post. The users specify the "start run" to start at time zero. For the "end run", the users specify its start time to be the length of the simulation run.

Sawmills exhibit rather high down times due to a number of reasons, including machine breakdown, operator breaks, cleaning of waste material, blockage, and lack of material to process [Aune and Leftbvre, 1975]. The process machinery needs periodic changes of saw blades and other maintenance. It is a necessary interruption in order to properly continue the process. It is planned ahead of time and carefully executed so that there will be no destructive interruption to the material process. Another type of down time is due to machine breakdowns. It is unpredictable in time and location, interrupts the on-going processes, and destroys the material being processed.

Preemptive schedules with random start times and durations can simulate the unpredictable machine breakdown; non-preemptive schedules with less variant start times and durations can simulate lunch breaks, maintenance, etc. The schedule database provides a schedule with random distributions for start time and duration. The users specify the parameters for the distributions. When the schedule database is loaded, random samples are generated from the distributions to make actual start time and duration for the schedule. The schedule is then posted in the message post of the system, waiting to be executed.

Process

Process events are responses to a process logic to be performed on an entity. Process events only apply to process nodes. Five types of process events can be applied to a process node that represent the operation sequence of "load - position - saw - unload - return":

Load — "Begin load" and "end load" pairs complete the load operation. When "begin load" starts, it sets the process node to busy state, schedules the "end load" event at a random time delay sampled from the load time distribution. When "end load" starts, it records statistics on the load operation, and posts the "begin operation" message to start the next operation in the sequence.

Position	Similar to the load operation, position operation has "begin position" and "end position" events. "End position" posts a "start saw" message for the process node.
Saw	Similar to the load operation, the saw operation has "begin saw" and "end saw" events. When "end saw" starts, it generates all new entities specified in process logic and sends them to the output node (usually a transfer node). It then examines the process logic for the remaining old entity for any new operations. If the next process logic for the old entity is at the same type of process node, it posts a "begin return" message for the process node. Otherwise, it posts a "begin unload" message for the process node.
Return	There are "begin return" and "end return" events for the return operation. The "end return" posts "begin position" for the process node.
Unload	"Begin unload" and "end unload" are the two events for unload operation. The "end unload" sends the old entity to the output of the process node, sets the node to idle state, and sends the message "ready to receive" to the input of the process node (usually a queue node).

Not all processor nodes use all five operations. Edgers, for example, do not use the return operation. The edgers have one pass operations, therefore, no entities are moved back for the second operation. The operation sequence for each process node is specified in the processor database. The process times for each operation are also specified as random distributions in the database.

III.4. Data requirement

S3 uses relational database data input / output. All files are stored in dBASE IV format (Table 3). Currently, Q+E is used to display and modify these data files. Any applications capable of accessing and modifying database files in dBASE IV format can manage the databases of S3.

The input data required to construct a sawmill model and run the simulation is large. Through data normalization and logical grouping, several relational database files serve as the input database for S3. These are predefined dBASE IV data files. Their names and data structure are recognized inside S3 and must be modified. The following sections explain these data files in greater details.

Before running a simulation model, S3 reads all required data from the database files. Whenever the users request a system report, S3 generates a statistics report as a database file that contains a snapshot of the system at the time of the report. A number of database filters are designed to allow the users to view the report database in a more selective manner. The report database file, and its filters, are discussed in sections on "report database".

Table 3. Database files for S3

Database	File name
Network database	network.dbf
Log distribution database	logpool.dbf
Process database	process.dbf
Processor database	processor.dbf
Connector database	connector.dbf
Queue database	queue.dbf
Transfer/collector database	transfer.dbf
Schedule database	schedule.dbf
Report database	report.dbf

III.4.A. Network database

The network database defines the configuration of the user specified sawmill model. Each record defines a node of the network. A network data file has at least two nodes: the source node and the sink node.

The network database defines these data fields for each node: name, type, group, input, and up to four outputs. The node name is the unique identification of the node in the system. The node type is the class name of the node, such as "board edger". S3 creates an object as an instance of this class. The grouping resembles the notion of a "machine center". If a node belongs to a certain group, it signifies that the node is a part of a "machine center" this group represents.

The network database uses input and output fields to define connections among nodes. The input field specifies the name of the node that serves as the input node. Similarly, the output field specifies the name of the output node. If a node is a transfer node, it can specify up to three more output nodes. The input field of a collector node need not be specified.

It is the users' responsibility to ascertain that the node names are unique, the node types are correct, and the connections are logical to accurately represent the sawmill layout the users intend to model. An example of the network database file is in the chapter on "System application".

III.4.B. Log distribution database

The log distribution database provides raw material distributions for log attributes. The log pool defined in network retrieves the records and builds random distributions for each of the log attributes. The log distribution database has data fields defined for species, grade, diameter, length, and taper (Table 4). The species and the grade are text strings up to 32 characters long each. The diameter is for the small end of log. The units of measure are inches for diameter, feet for length, and inches per 16 feet for taper. The cumulation field contains the cumulative probability for the combinations of the log attributes specified in the record. The cumulative probability values are entered as percentages. If the database file predefines a log batch, the cumulative field contains the actual number of logs with the specified attributes.

Table 4. Probability entry for the log distribution database file

SPECIES	GRADE	DIAMETER	LENGTH	TAPER	CUMULATION
df					40%
hem					100%
		6			0%
		12			20%
		15			40%
		20			80%
		25			100%

To simulate random log distributions for each log attribute, the users specify one parameter field for each record, and enter the cumulative probability corresponding to the attribute (Table 4). The range of cumulative probability must be from 0% to 100%. For example, in Table 4, the probability is 40% (0% to 40%) for species "df", and 60% (40% to 100%) for species "hem".

To simulate predefined log batches, users specify a combination of log attributes for each record and enter the number of logs with those attributes in the cumulation field (Table 5). S3 retrieves these predefined distributions from the database file and creates a collection of log entities with the specified attributes and piece counts before the simulation begins. During the simulation, whenever a log is requested, a log is removed from the collection. The simulation ends when there are no more logs in the collection.

Table 5. Piece entry for the log distribution database file

SPECIES	GRADE	DIAMETER	LENGTH	TAPER	CUMULATION
df	sawlog1	6	8	0	20
df	sawlog2	12	12	2	30
hem	mill	10	14	2	65
hem	mill	12	18	3	14

III.4.C. Process database

The process database stores the process logic for log breakdown. A process logic, identified by its process tag, is a sequence of operations. Each sequence contains an operation tag, the type of node the operation is performed on, information on the old entity and on the new entities generated. The data fields for the process database are discussed in detail below:

Process tag — This tag is a unique identification for the process. A process logic is accessed through its process tag.

Operation tag — This tag uniquely identifies the operation within the process. An operation references other subsequent operations through the their tags.

Node type — This field specifies the type of node to perform the operation. During simulation, S3 tries to locate a node of the specified type to perform the operation. S3 may also assign a node of compatible types, if nodes of the exact type are not available.

Old entity — Four fields specify the properties of the remaining entity, including its type and dimension. S3 uses three fields to specify the dimension of an entity (see sections on "entity" for more detail).

Old tag — This field specifies the tag of the operation to apply to the old entity. If it is not specified, the entity needs no more operations, as for lumber.

New entity — Data fields for new entity are similar to that for the old entity. In addition, the replication field specifies the number of replicates for the new entity. Replicated entities are of the same type and dimension, and use the same operations for further processing. If new entities are not the same in type or dimension, or require different operations, the "more tag" field specifies the operation to generate another batch of the new entities.

New tag — Similar to the old tag, this field specifies the operation to apply to the new entities.

More tag — This field stores the tag of the operation to generate more new entities. This operation is a dummy operation that serves to generate entities of difference attributes from the ones specified in "new entity" field. This operation does not consume time.

An example the of process database file is presented in the chapter on "System application".

III.4.D. Processor database

The processor database contains the details for all the processor nodes defined, including processor type, operation types, and process times. A processor database may contain more nodes than used in a network. Only the nodes specified in the network database are used in the simulation. The database acts like a library of processor node that the users can build sawmill models from.

Name	The name field stores the name of the processor node that uniquely identifies the node in the network.
Description	This field provides space that the users can use for additional text description.
Type	The type of a processor node can be any of the classes inherited from the class processor (see the section on "Node" for more detail).
Operation	This field contains a collection of the operation types. S3 retrieves the collection, and then creates process time distributions from each of the operation types in the collection. Valid operation types are: load, position, saw, return, and unload. A node, such as a resaw, can define all five. Other nodes may define fewer operation types. For example, the board edgers only define load, saw, and unload operations.
Process time	Two data fields define the process time for an operation type. These are the two parameters for a random distribution. Each processor node defines process times for up to five operations. Process times are measured in seconds.

III.4.E. Connector database

The connector database stores data for a connector. The input and output nodes are specified in the network database. The data fields specific to connector databases are discussed below:

Type	A connector can be a belt conveyor, a roller conveyor, or a forklift. It is actually the name of a class defined in S3.

Distance — The distance of a connector is the distance between the two nodes the connector links. It is not necessarily the shortest linear distance between two points, because of layout constraints. The distance usually remains unchanged during a simulation. The distance is measured in feet.

Speed — The distance and speed together provide the travel time for the connector. The speed is a factor that the users can adjust during a simulation to change the travel time in the connector. The speed is measured in feet per minute.

Capacity — This is mainly used to specify the load size of a forklift. It does not apply to continuous conveyors like belt and roller conveyors. The capacity is measured in number of pieces.

III.4.F. Queue database

The queue database specifies data fields for the queue nodes. As with any other node database, it defines the name and the description fields. The queue database defines two more data fields specific to the queue nodes. The capacity field specifies the capacity of the queue, measured in number of pieces. The reasons for not using the physical dimensions of entities as the bases for capacity are that entities tend to overlap while stored in a queue, and that the overlap behavior is hard to quantify. The fixed capacity field specifies whether the queue has a fixed or flexible capacity. A queue node with fixed capacity blocks incoming entity when it reaches its capacity. When the capacity is reached, a queue node with flexible capacity allows itself to expand by entering an "overflow" state, while still accepting incoming entities.

III.4.G. Transfer/collector database

The transfer/collector database contains records for transfer and collector nodes. This database defines the fields: name, description, and criterion. The criterion is for transfer node only. The connections from a node to other nodes are specified in the network database. Each transfer node can have different routing strategies. The method implemented in S3 is discussed in the section on "Transfer". For a collector node, this database is used to enter additional text descriptions.

III.4.H. Schedule database

The schedule database contains information on events that affect the schedule of the operation, such as start / stop times for the run, and start / stop times for a node to be down. These start and stop time pairs are read into the system, then messages corresponding to these events are generated and placed in the message post before the simulation is started. The data fields specific to schedule database are discussed below:

Machine — The machine is the name of the target node for this event to act on. If it is not specified, the event applies to the whole network, such as the "run" event.

Start time — Start time is a distribution specified by two parameters, the mean and the standard deviation. S3 generates a random sample from the distribution to produce a time. The users can set the standard deviation to zero for deterministic times. The start time is measured in seconds.

Duration — Similar to the start time, the duration is also a distribution specified by two parameters, the mean and the standard deviation. The duration time is measured in seconds.

Preemptive — Machine downtime events are preemptive, while events such as lunch break and saw change do not have to be preemptive. S3 defers the execution of non-preemptive events until the current event is finished.

III.4.I. Report database

The system output from the simulation is saved into a single database. Selective SQL views into the database are provided to display the output in different forms. These view filters include raw material used, final products generated, entity counts at each machine center, states of machine centers, and utilization of machine centers.

Process view — Each process node displays statistics on the operations it has undergone. For example, the process view for a resaw displays the statistics on its process times for load, position, saw, return, and unload.

Throughput view — Statistics on system throughput are displayed from the throughput view. Each node in the model produces statistics on linear, volume, and piece throughput.

Entity flow view Piece flow by entity types in each node is displayed from the entity view. The total numbers of entities of different types are tallied for each node.

Utilization view Statistics on system utility are displayed from the utility view. The process nodes produce utility statistics on idle, busy, and down times. The queue nodes produce utility statistics on queue size, waiting time, and times the queue is in the states of empty, in use, full, or overflow.

Input view Input view provides log tally by species, grade, diameter, length, and taper.

Output view Output view provides lumber tally by species, grade, length, width, thickness, and cross section.

Examples of these output database files and views are presented in the chapter on "System application".

IV. SYSTEM APPLICATION

In this chapter, an attempt is made to walk the users through the steps of constructing a sawmill model, running the model, and presenting the output. Since S3 is a file-based application, it is only natural to discuss the setup of data files first, and then go through a simulation session using these data files.

IV.1. Machine centers

IV.1.A. Sawmill layout

The sawmill layout consists of each of the major process node types. The concept of "machine center" is used to describe the nodes involved in the sawmill layout. Typically, a machine center has a collector node, a queue node, a process node, and a transfer node. The collector node handles the multiple input connections, while the transfer node handles the multiple output connections. Machine centers are connected by connectors. Inside a machine center, nodes are linked directly without use of connectors. A graphical representation of a typical sawmill layout is shown in Figure 4. The specific machine centers in Figure 4 are:

Log yard — The log yard consists of the source node and a transfer node. The source node generates log entities and sends them directly to the transfer node where the entities are distributed to the down stream machine centers connected through roller conveyors.

QB head saw — The head saw machine center is a combination of a collector node, a buffer deck, a quad band head saw, and a transfer node. It receives log entities from the log yard and sends entities to the resaw, the cant edger, the trimmer/sorter, and the chipper.

DB head saw — This head saw machine center is a combination of a collector node, a buffer deck, a double band head saw, and a transfer node. It receives log entities from the log yard and sends entities to the resaw, the cant edger, the trimmer/sorter, and the chipper.

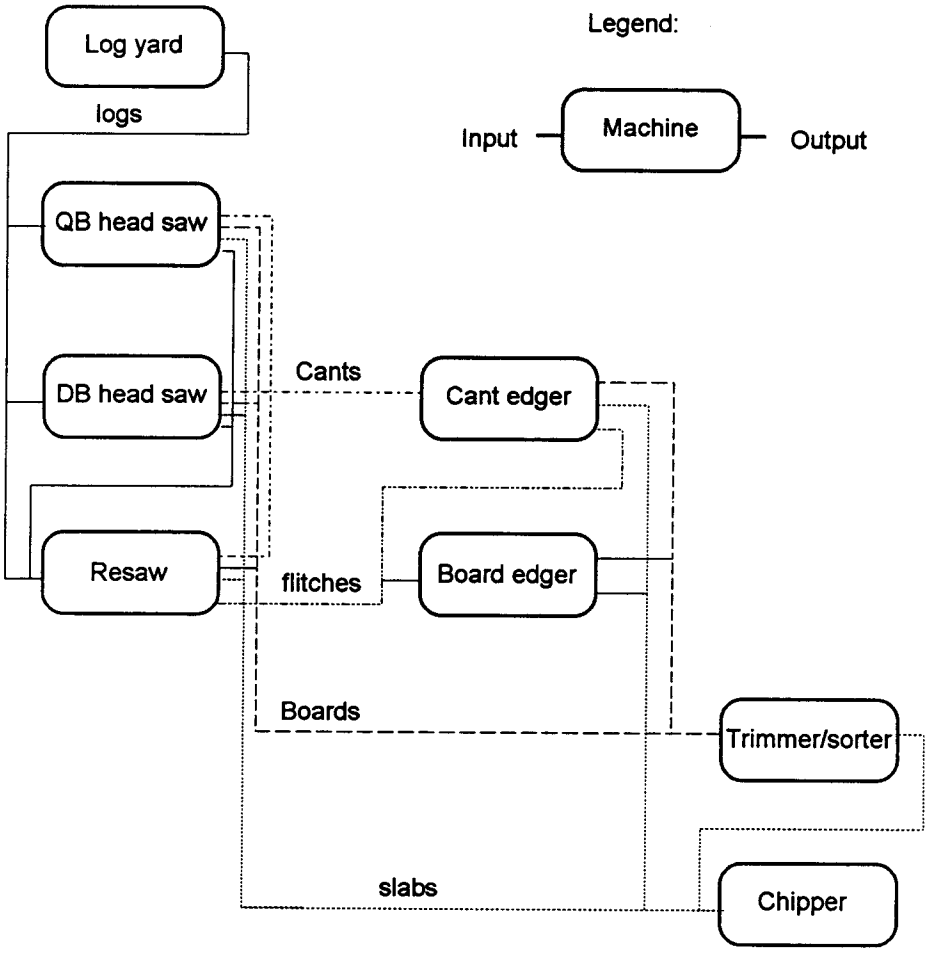


Figure 4. Sawmill layout

Resaw — This machine center consists of a collector node, a buffer deck, a single band resaw, and a transfer node. It receives entities from the two head saws up stream and sends new entities downstream to the cant edger, the board edger, the trimmer/sorter, and the chipper.

Cant edger — This machine center contains a collector node, a buffer deck, a multiple saw cant edger, and a transfer node. It is connected up stream to the DB head saw, the QB head saw, and the resaw. Its output entities go to the board edger, the trimmer/sorter, and the chipper.

Board edger — The board edger machine center contains a collector node, a buffer deck, a multiple saw board edger, and a transfer node. It receives board entities from the DB head saw, the QB head saw, the resaw, and the cant edger. Its output entities go to the trimmer/sorter and the chipper.

Trimmer/sorter This machine center consists of a trimmer node followed by a sorter node. No collector node is needed since the trimmer node is a collector node. No buffer deck is used in this machine center, because process times are not the concern here. The main function of this machine center is to record the lumber tallies. This machine center receives board and lumber entities from all up stream machine centers and sends all entities to the sink node.

Chipper This machine center contains a chipper node. It receives slab entities from all up stream machine centers, tallies the entities, and then sends them to the sink node.

IV.1.B. Network

All the nodes mentioned in the machine centers are specified in the network database file (Table 6). These nodes include collectors, queues, processors, transfers, and connectors. The network database file specifies the connection among the nodes. Nodes inside a machine center connect directly, without the use of connectors. Connectors are used to connect one machine center to another.

Table 6. Network database file for application example

NAME	TYPE	GRP	INPUT	OUTPUT	OUT2	OUT3	OUT4
BE-saw	BoardEdger	BE	BE-deck	BE-sort			
BE-coll	Collector	BE		BE-deck			
BE-deck	Queue	BE	BE-coll	BE-saw			
BE-sort	Transfer	BE	BE-saw	Trimmer	Chipper		
CE-BE-belt	Belt	CE	CE-sort	BE-coll			
CE-saw	CantEdger	CE	CE-deck	CE-sort			
CE-coll	Collector	CE		CE-deck			
CE-deck	Queue	CE	CE-coll	CE-saw			
CE-sort	Transfer	CE	CE-saw	CE-BE-belt	Trimmer	Chipper	
DH-CE-belt	Belt	DH	DH-sort	CE-coll			
DH-RS-belt	Belt	DH	DH-sort	RS-coll			
DH-BE-belt	Belt	DH	DH-sort	BE-coll			
DH-coll	Collector	DH		DH-deck			
DH-saw	DBHeadSaw	DH	DH-deck	DH-sort			
DH-deck	Queue	DH	DH-coll	DH-saw			
DH-sort	Transfer	DH	DH-saw	DH-RS-belt	DH-CE-belt	DH-BE-belt	Chipper
LK-DH-roll	Roller	LK	LK-sort	DH-coll			
LK-QH-roll	Roller	LK	LK-sort	QH-coll			
LK-RS-roll	Roller	LK	LK-sort	RS-coll			
LK-sort	Transfer	LK	Source	LK-DH-roll	LK-QH-roll	LK-RS-roll	
Chipper	Chipper	NET		Sink			
Sink	Sink	NET					
Source	Source	NET		LK-sort			
QH-CE-belt	Belt	QH	QH-sort	CE-coll			
QH-RS-belt	Belt	QH	QH-sort	RS-coll			
QH-BE-belt	Belt	QH	QH-sort	BE-coll			
QH-coll	Collector	QH		QH-deck			
QH-saw	QBHeadSaw	QH	QH-deck	QH-sort			
QH-deck	Queue	QH	QH-coll	QH-saw			
QH-sort	Transfer	QH	QH-saw	QH-RS-belt	QH-CE-belt	QH-BE-belt	Chipper
RS-CE-belt	Belt	RS	RS-sort	CE-coll			
RS-BE-belt	Belt	RS	RS-sort	BE-coll			
RS-coll	Collector	RS		RS-deck			
RS-deck	Queue	RS	RS-coll	RS-saw			
RS-saw	SBResaw	RS	RS-saw	RS-sort			
RS-sort	Transfer	RS	RS-saw	RS-BE-belt	RS-CE-belt	Trimmer	Chipper
Sorter	Sorter	TS		Sink			
Trimmer	Trimmer	TS		Sorter			

IV.1.C. Processor

Table 7 shows the process database file that lists the attributes for processor nodes. The connection from a processor node to other nodes has already been specified in the network database file. This file specifies the process type and process times for the processor nodes. The process times are specified in pairs of mean and standard deviation. S3 loads this file and uses a

normal distribution for each pair of process time parameters specified by the users. The process times are measured in seconds.

Table 7. Processor database file

NAME	DESC	TYPE	PROCESS
DH-saw	Double band head saw	DBHeadSaw	load position saw unload return
QH-saw	Quad band head saw	QBHeadSaw	load position saw unload return
RS-saw	Single band resaw	SBResaw	load position saw unload return
CE-saw	Cant edger	CantEdger	load saw
BE-saw	Board edger	BoardEdger	load saw

NAME	LOAD		POSTITION		SAW		UNLOAD		RETURN	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
DH-saw	10	3	10	3	10	3	5	0	5	0
QH-saw	10	3	10	3	10	3	4	1	2	0
RS-saw	20	4	16	4	10	4	10	3	5	2
CE-saw	10	2			15	5				
BE-saw	15	5			10	2				

IV.1.D. Queue

Table 8 lists all the queue nodes defined for the sawmill layout. Aside from the connection information already defined in network database file, each queue needs to know its capacity and whether it is allow to expand (the "FIXEDCAP" field). The capacity is measured in number of entities. The "FIXEDCAP" field accepts Boolean values.

Table 8. Queue database file

NAME	DESC	CAPACITY	FIXEDCAP
QH-deck		5	N
DH-deck		10	N
CE-deck		50	N
RS-deck		50	N
BE-deck		50	N

IV.1.E. Transfer/collector

S3 currently only defines one criterion for transfer station, "byProcess". The main purpose for this database file is to enter additional text about each of the nodes (Table 9).

Table 9. Transfer/collector database file

NAME	DESC	TYPE	CRITERION
BE-sort		Transfer	byProcess
CE-sort		Transfer	byProcess
RS-sort		Transfer	byProcess
DH-sort		Transfer	byProcess
QH-sort		Transfer	byProcess
LK-sort		Transfer	byProcess
RS-coll		Collector	
CE-coll		Collector	
BE-coll		Collector	
QH-coll		Collector	
DH-coll		Collector	
Trimmer		Trimmer	
Sorter		Sorter	
Chipper		Chipper	

IV.1.F. Connector

The connector database specifies the parameters for each connector defined in the sawmill layout (Table 10). The distance is specified in feet and the speed in feet per minute.

Table 10. Connector database file

NAME	DESC	TYPE	DISTANCE	SPEED
LK-QH-roll		Roller	200	15
LK-DH-roll		Roller	200	15
QH-CE-belt		Belt	200	20
QH-RS-belt		Belt	200	20
DH-CE-belt		Belt	300	20
DH-RS-belt		Belt	300	20
RS-CE-belt		Belt	400	20
RS-BE-belt		Belt	400	20
CE-BE-belt		Belt	400	20
LK-RS-roll		Roller	200	15
QH-BE-belt		Belt	200	20
DH-BE-belt		Belt	300	20

IV.2. Log distribution

The log distribution is defined in Table 11. When S3 loads this file, it creates empirical distributions for each log attribute field. Step distributions are created for species and grade, because they are symbolic attributes. Line-distributions are created for diameter, length, and taper, because they are numerical attributes. To generate a log, S3 takes a random sample for

each of the attributes and assigns them to the new log. Interdependencies among attributes (diameter and length, for example) are not incorporated in the current version of S3.

Table 11. Log distribution database file

SPECIES	GRADE	DIAMETER	LENGTH	TAPER	CUMULATION
df hem					40% 100%
	sawlog1 sawlog2 mill				20% 60% 100%
		6 12 15 20 25			0% 20% 40% 80% 100%
			7 10 15 20 25 30		0% 10% 30% 50% 80% 100%
				0 1 2 3	0% 40% 80% 100%

IV.3. Process logic

The sawing pattern used in this example is based on the "best opening face" method. It generates a process logic that uses the quadruple band head saw for the first breakdown, single band resaw for the second breakdown, cant edger for cant breakdown, and board edger for flitch processing. Figure 5 illustrates the sequence of operations in this sawing pattern.

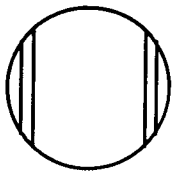
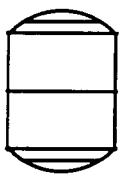
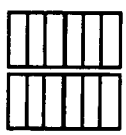
The log is 16.8 inches in diameter. A quadruple band head saw breaks down the log into a cant with 12 x 16.8 inches in cross section, two 1 inch thick flitches, and two slabs. The flitches are edged to 1 x 8 inches cross section.	
A single band resaw breaks down this cant into two cants with 6 x 12 inches cross section, two 1 inch thick flitches, and two slabs. The flitches are edged to 1 x 8 inches cross section. The operation is performed in five passes by the single band resaw.	
A multiple saw blade cant edger breaks down each 6 x 12 inches cant into six 2 x 6 inches boards.	

Figure 5. The sawing pattern for log process logic in S3

Table 12 is the process data file that defines the log process logic based on the sawing pattern in Figure 5. Seven operations are defined that require a quad band head saw, a single band resaw, and a cant edger.

Table 12. The process database file

PROC	SEQ		NODETYPE	OLDTYPE	OLDX	OLDY	OLDZ	OLDSEQ
0	0		QBHeadSaw	Cant	14	16	10	-1
0	1		QBHeadSaw	Cant	12	16	10	2
0	2		SBResw	Cant	12	15	10	3
0	3		SBResw	Cant	12	14	10	4
0	4		SBResw	Cant	12	13	10	5
0	5		SBResw	Cant	12	12	10	6
0	6		SBResw	Cant	12	12	10	-1
0	7		CantEdger	Cant	6	12	10	-1

PROC	SEQ	DUP	MORESEQ	NEWTYPE	NEWX	NEWY	NEWZ	NEWSEQ
0	0	2	1	Slab	0	0	0	-1
0	1	2	-1	Flitch	1	8	10	-1
0	2	1	-1	Slab	0	0	0	-1
0	3	1	-1	Flitch	1	8	10	-1
0	4	1	-1	Slab	0	0	0	-1
0	5	1	-1	Flitch	1	8	10	-1
0	6	2	-1	Cant	6	12	10	7
0	7	6	-1	Board	2	6	10	-1

In Table 12, the "PROC" field identifies the process tag and the "SEQ" field identifies the operation tag. The "NODETYPE" field identifies the type of processor node the operation requires. The "OLDTYPE" is the type of remaining entity after the operation. The "OLDX", "OLDY", "OLDZ" specify the dimension of the entity (The meanings of the three dimensions change with the entity type. See sections on "Entity" for more detail). The "OLDSEQ" field identifies the tag of the operation to be applied to the remaining entity. The value of "-1" means no operation.

The lower half of Table 12 specifies data on new entities as a result of the operation. Most fields correspond to the upper half of the table. The "DUP" field specifies the number of identical entities. For example, the operation tag "0" generates two identical slabs. Each operation can only specify one entity type and dimension for the new entities. Since some operations can produce multiple non-identical entities, the "MORESEQ" field is used to associate a special operation whose sole purpose is to list those new entities. For example, operations "0" and "1" together specify a quadruple sawing operation that produces two slabs and two flitches.

IV.4. Event Schedule

Two schedules are defined for this model (Table 13). Schedules are posted into the message post before the simulation starts. The first schedule, "Run", starts and stops the simulation. When S3 loads the "Run" schedule in specified Table 13, it creates two messages: "beginRun" to be invoked at time 0 second; "endRun" to be invoked at time 3600 seconds. Because the "Run" schedule is preemptive, the "endRun" stops the simulation regardless of other messages waiting in the message post.

Table 13. Example schedule database file

NAME	DESC	MACHINE	S MEAN	S STD	D MEAN	D STD	PREEMPT
Run			0	0	3600	0	Y
Down		BE-saw	1800	600	60	10	Y

The second schedule "Down" applies to the node named "BE-saw". When S3 loads this schedule, it creates the "beginDown" and "endDown" messages and posts them to the message post. The activation time for "beginDown" message is a random value sampled from a normal distribution specified by mean of 1800 seconds, and standard deviation of 600 seconds. When the node "BE-saw" receives this message, it changes itself into down state, and since the message is specified as preemptive, it sends the entity it is currently processing to the sink node, essentially "trashing" the entity. The duration for the down time is a random value from a normal

distribution specified by a mean of 60 seconds and a standard deviation of 10 seconds. The activation time of "endDown" message is the time of "beginDown" plus the duration.

IV.5. User interface

S3's approach to a user interface is simple and straightforward. A simulation is started by executing the S3 program from Microsoft Windows. The main window of S3 (Figure 6) is displayed on the screen. The size and position of the window can be rearranged by the users. The title bar shows the name of the application. The menu bar of the window provides action choices for the users. The display area ("client area" of the window, in Microsoft Windows' jargon) displays text that shows the results of actions invoked by the user. The sections below illustrate the menu items and their actions.

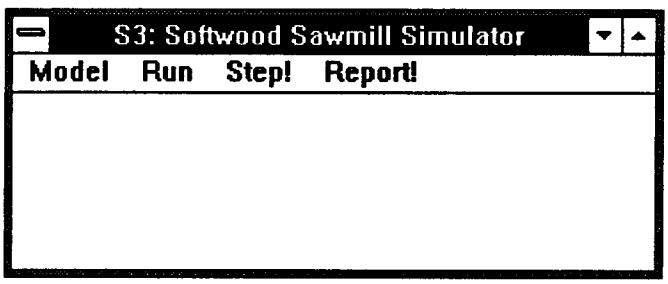


Figure 6. User interface for S3

IV.5.A. Model

Selecting the "Model" menu on the menu bar will pull down a sub menu from which users can load a sawmill model, or exit from the program (Figure 7).

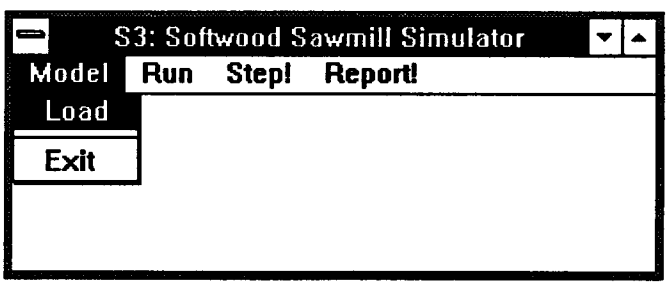


Figure 7. The "Model" menu in S3

When the users select the "Load" item, S3 loads a predefined sawmill model. S3 will first load the network database, then create and load data for individual machine centers specified in the network, and finally connect these machine centers. The display area of the main window will show the progress of data file loading. Once the model is loaded, the sawmill model is ready to run.

To exit from S3, the users can select the menu item "Exit". The trace text file, if defined, will close itself. S3 will then exit to the Microsoft Windows environment.

IV.5.B. Run

Once a model is properly loaded, S3 can execute a simulation run on the users' command. Selecting the "Run" menu item on the menu bar will pull down a sub menu from which users can start, stop, continue, or reset the simulation run (Figure 8). The users can step through each event in the simulation by repeatedly selecting the "Step!" item on the menu bar.

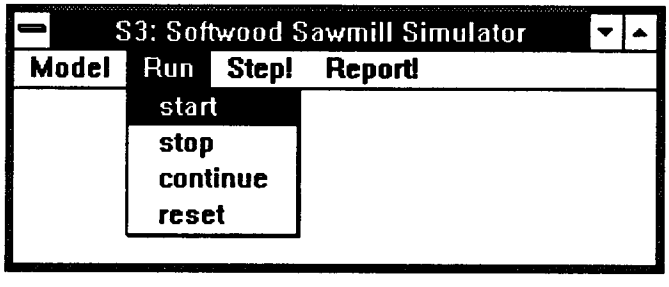


Figure 8. The "Run" menu in S3

When the users select the "start" item on the "Run" sub menu, S3 will clear all system statistics, remove all events and entities, set the states of machine centers to idle, set the system clock to time zero, reload all the events defined in the schedule database, and start the message loop. One of the required events loaded from schedule is "begin run" which will be executed first, dispatching the message "generate log" to the source node of the network. The model will then start a chain reaction of event execution and generation. The display area of the main window will display each event as messages are dispatched and posted. S3 will continue running the simulation, until there are no more messages to dispatch in the system or when the users select the "Stop" menu item.

On the selection of the "Stop" menu item, S3 suspends execution by going into a sleep state, thus no new event will be executed. The users can awaken this sleep state by selecting the "Continue" menu item which will make S3 enter the execution state again.

S3 can enter the sleep state in response to other user actions. When users select the "Step!" menu item, S3 will first enter the sleep state if not already so, then dispatch one message and stop. The users can repeatedly select the "Step!" item to run the simulation event by event.

On the selection of the "Reset" menu item from the "Run" sub menu, S3 clears all statistics contained in the system. This is useful in situations when the users are only interested in the "steady state" behavior of the system and do not desire statistics collected for transient states in the system.

IV.5.C. Report

At any time during or after the simulation run, a snapshot of the system state can be saved by the selection of the "Report!" item from the menu bar (Figure 8). On selection of the "Report!" item, S3 temporarily suspends the event execution, and then saves statistical observations for each machine center and the system itself to a database file predefined as "report.dbf". This reporting can be repeated as often as the users wish. Each time the users ask for a report, the database file is cleared of all previous records, then new records are appended to the same database file. Details about the report and database file are discussed in the section on "Output data", and in the section below on "Simulation output".

IV.6. Simulation output

The system output from the simulation is saved into a dBASE file called "report.dbf". This is a predefined database file. Selective SQL views into the database are provided to display the output in different forms. These view filters include raw material used, final products generated, entity counts at each machine center, states of machine centers, and utilization of machine centers.

Table 14 shows the original record format for the output database. The individual fields are explained below:

Table 14. Record entries in report database file

DATE	TIME	NODE	TYPE	TALLY	STAT	ITEM	N	MIN	MEAN	MAX	STD
2-1-1993	1:00:00	DH-saw	DBHeadSaw	process	saw		49	2	9	16	19
2-1-1993	1:00:00	DH-saw	DBHeadSaw	entity	count	Cant	48				
2-1-1993	1:00:00	DH-saw	DBHeadSaw	throughput	linear		132	9	84	222	1554
2-1-1993	1:00:00	QH-deck	Queue	utility	wait		20	0	1	8	6

DATE This field stores the calendar date when the report is generated.

TIME This field stores the simulation time when the report is generated. This time is measured from the start of simulation.

NODE This field identifies the name of the node for which the statistics are generated in this record. The users may create a query that returns all statistics on a specific node using this field.

TYPE This field identifies the type of the node. Using this field, the users can create a view that contains all statistics for nodes of a particular type, for example, queue nodes.

TALLY This field identifies the type of tally as defined in the framework of S3. The tallies include statistics for entity, log, lumber, process, queue, throughput, utility, and volume. See the section on "Statistical observations" in Chapter III for more detail.

STAT This field identifies the counter defined in a tally. See the section on "Statistical observations" in Chapter III for more detail.

ITEM This field identifies the items in a bean counter. It is not used for other types of tallies and counters. See the section on "Statistical observations" in Chapter III for more detail.

N, MIN, MEAN, MAX, STD These fields contain the parameters for descriptive statistics. The "N" field stores the number of occurrences. The "MIN", "MEAN", and "MAX" fields store the minimum, average, and maximum of the values observed. The "STD" field stores the standard deviation of the observations. The bean counters use only the "N" field. The units of measure for these fields are specific to the types of tally and counter for a record.

In Table 14, the first record is a process tally for the node "DH-saw" that is a type of "DBHeadSaw". The statistics recorded for the tally is from a state counter for the "saw" operation. The second record is an entity tally for the same node. The recorded statistic is from the bean counter for the item "Cant". The third record is a throughput tally for the same node. The statistic recorded is the linear throughput. The fourth record is a queue tally for the node "QH-deck" that is a queue node. It records the statistics on the time entities spend in the queue.

Since the users may view this report database file using an application other than Q+E, the SQL view statements are provided, so that the users can use these statements directly if the application supports SQL query, or use them as templates for other types of database queries provided in the application. For more details, refer to The Whitewater Group [1991b] or other SQL references for more details about the SQL syntax. Because all statistics are recorded into a single database file, the number of records is rather large (260 records in the example model). The users should make certain that their application can handle a database of this size.

IV.6.A. System state

System states are displayed through the process view filter (Table 15). The SQL statement is "SELECT NODE, STAT, MIN, MEAN, MAX, STD, N FROM REPORT.DBF WHERE TALLY = 'process' ORDER BY NODE, STAT". Each process node displays statistics on the operations it has undergone. For example, the process view for a resaw displays the statistics on its process times for load, position, saw, return, and unload. All the values on process times are measured in seconds.

Table 15. Process view of system output

NODE	STAT	MIN	MEAN	MAX	STD	N
Single-band-resaw	load	0:00:04	0:00:05	0:00:07	0:00:03	4
Single-band-resaw	position	0:00:02	0:00:05	0:00:09	0:00:14	4
Single-band-resaw	return	0:00:02	0:00:04	0:00:09	0:00:07	14
Single-band-resaw	saw	0:00:03	0:00:09	0:00:18	0:00:29	18
Single-band-resaw	unload	0:00:00	0:00:01	0:00:04	0:00:04	3

IV.6.B. System throughput

Statistics on system throughput are displayed from the throughput view (Table 16). The SQL statement is "SELECT NODE, STAT, MIN, MEAN, MAX, STD, N FROM REPORT.DBF WHERE TALLY = 'throughput' ORDER BY NODE, STAT". Each node in the model produces statistics on

linear, volume, and piece throughput. The units of measure are feet per minute for linear throughput, pieces per minute for piece throughput, and board feet per minute for volume throughput.

Table 16. Throughput view of system output

NODE	STAT	MIN	MEAN	MAX.	STD	N
Board-edger	linear	83	272	384	5762	62
Board-edger	piece	10	14	17	4	62
Board-edger	volume	0.38	1.26	1.78	0.12	62

Piece flow by entity types in each node is displayed from the entity view (Table 17). The SQL statement is "SELECT NODE, ITEM, N FROM REPORT.DBF WHERE TALLY = 'entity' ORDER BY NODE". The total numbers of entities of different types are tallied for each node.

Table 17. Entity view of system output

NODE	ITEM	N
QBHR-deck	Log	23
QBHR-drop-sorter	Cant	18
QBHR-drop-sorter	Flitch	38
QBHR-drop-sorter	Slab	38
QB-headsaw	Cant	18
QB-headsaw	Flitch	38
QB-headsaw	Slab	38

IV.6.C. System utility

Statistics on system utility are displayed from the utility view (Table 18). The SQL statement is "SELECT NODE, STAT, MIN, MEAN, MAX, STD, N FROM REPORT.DBF WHERE TALLY = 'utility' ORDER BY NODE, STAT". The process nodes produce utility statistics on idle, busy, and down times. The queue nodes produce utility statistics on queue size, waiting time, and times the queue is in the states of empty, in use, full, or overflow. All time-wise statistics are measured in seconds. The size of the queue is in number of pieces.

Table 18. Utility view of system output

NODE	STAT	MIN.	MEAN	MAX.	STD	N
Board-edger	busy	0:00:03	0:00:08	0:00:18	0:00:28	32
Board-edger	down	0:01:02	0:01:02	0:01:02	0:00:00	1
Board-edger	idle	0:00:03	0:00:09	0:00:23	0:00:24	33
Board-edger-deck	empty	0:00:00	0:00:00	0:00:00	0:00:00	1
Board-edger-deck	size	0	7	12	10	76
Board-edger-deck	use	0:00:00	0:00:21	0:01:16	0:12:48	75
Board-edger-deck	wait	0:00:00	0:00:21	0:01:16	0:12:33	76

IV.6.D. Input statistics

Input statistics are displayed from the log view (Table 19). The SQL statement is "SELECT STAT, ITEM, N FROM REPORT.DBF WHERE TALLY = 'log' ORDER BY STAT, ITEM". It is basically a log tally by species, grade, diameter, length, and taper. The units of measure are inches for diameter, feet for length, and inches per 16 feet for taper. The species and grades are text strings.

Table 19. Log view of system output

STAT	ITEM	N
diameter	10	3
diameter	11	2
diameter	12	2
diameter	13	1
diameter	14	2
diameter	16	1
diameter	17	6
diameter	18	6
diameter	19	10
grade	mill	15
grade	sawlog1	10
grade	sawlog2	18

IV.6.E. Output statistics

Output statistics are displayed from the lumber view (Table 20). The SQL statement is "SELECT NODE, TALLY, STAT, ITEM, N FROM REPORT.DBF WHERE TALLY = 'lumber' ORDER BY NODE, STAT". Lumber view provides lumber tally by species, grade, length, width, thickness, and cross section. The reason for putting the node name in the view is that a sawmill model may have more than one sorter node. The units of measure are feet for length, and inches for width and thickness. The cross section is a literal string, similar to the grade and species.

Table 20. Lumber view of system output

NODE	TALLY	STAT	ITEM	N
Sorter	lumber	species	df	55
Sorter	lumber	species	hem	58
Sorter	lumber	thickness	1	32
Sorter	lumber	thickness	2	81
Sorter	lumber	width	8	32
Sorter	lumber	width	6	81
Sorter	lumber	x section	1x8	32
Sorter	lumber	x section	2x6	81

IV.6.F. System summary

Table 21 shows a summary report for the model defined in Table 6. This table sumarizes the system input and output, utilization of processor nodes, and statistics on queue nodes. The results are based on the report at four hours and thirty minuets of the simulation.

Table 21. System summary for application example

Queue node	Utilization
BE-deck	76%
CE-deck	0%
DH-deck	0%
QH-deck	0%
RS-deck	70%

Processor node	Utilization
BE-saw	50%
CE-saw	42%
DH-saw	30%
DH-saw	33%
RS-saw	50%

Entity	Count
Log generated	471
Log processed	470
Lumber	3067
Slab	1656
Flitch	682
Cant	1385
Board	3067

The utilization of a queue node is the ratio of the average size of the queue to the queue capacity. The utilization of a processor node is the ratio of the cumulative times the processor node is busy to the time of simulation.

The system input is log. The system output is lumber. Cants, flitches, slabs, and boards are intermediate entities generated during the conversion of log into lumber.

V. CONCLUSIONS

V.1. Achievements

The primary objective of S3 is to provide the basic functional modules required for developing sawmill simulation models. This is accomplished by developing a library of objects (classes) representing the primary components of a sawmill. S3 classes are easy to comprehend, modify and expand, thus making it possible to model a wide variety of sawmill systems.

The choice of an object-oriented environment as the platform for S3 offers several advantages. The classes defined in S3 are readily reusable and easily expandable, and can be ported to other object-oriented environments.

The functionality for a discrete event simulation is basically complete in S3. The use of the object-oriented design approach makes the process of simulation intuitive and straightforward.

S3 offers a simple and intuitive user interface for running and testing the simulation model. Because Actor is an interpretative environment, its built-in debugging facilities can be readily utilized for model testing.

S3 simplifies the separation of functionality between log breakdown and log throughput by treating log breakdown process logic as external data input. This allows integration of log process logic developed with more powerful optimization techniques.

The use of an industry standard format for the databases used for both input and output data provides easy integration to other applications. The output from simulation can be post-processed in other applications with more elaborate report and display capabilities.

V.2. Limitations and enhancements

The limitations in S3 result from two sources: (1) the design boundaries for the current version of S3 and (2) the limitations of the Actor programming environment. Whereas the first set of limitations can be addressed by enhancing S3, the second cause would require the conversion of S3 into a more efficient programming environment.

V.2.A. Design boundaries

S3 was developed to target the operation of Northwest softwood dimension lumber manufacturing. The process and transfer stations contain embedded process logic specific to dimension lumber manufacturing. For example, the transfer stations may have up to four outputs. These limits can be expanded to represent a different system.

The trimmer and sorter machine centers are not modeled as processor nodes, because of their simple operations. These two node types can be enhanced to processor nodes so that their operations can be modeled in greater detail.

S3 generates logs by taking random samples from several log attribute distributions and assembling them into a log. S3 keeps one distribution for each log attribute, to reduce the requirement on storage and data entry. More accurate representations require that correlation between log attributes be modeled in the system.

The process times at process stations are not linked to the properties of material being processed, because of the difficulties in establishing the relationship between the process time and the material attributes. Instead, the process times are random times taken from user specified distributions. Again, more accurate representations would require that process times be specified as a function of material attributes.

S3 treats the log process logic as static data; operation sequences and operation stations assigned to the sequences do not change during the simulation. Thus, the dynamics of a sawmill cannot be completely modeled. More intelligent decisions at transfer stations would require embedding an "expert system" component into the system.

S3 only keeps track of entity piece counts in a queue, without regard to the entity size and overlap, because of the difficulties in quantifying the overlap factors for different types of entities and operation conditions. As a result, the queue sizes may not be the most accurate representations of surge decks in a sawmill.

S3 currently retrieves data from database files made by the users without any integrity or validity checking. A much desired enhancement to the current implementation would be the validation of user data. The network database should be checked against the illogical connection of stations; the log distribution database should be validated against incorrect distributions; and the process database should be validated against incorrect sequences and station requirements.

Graphical representation of sawmill layout, log distribution, and process pattern would greatly improve the users' ability to both validate the data and comprehend the process. Graphical representation of output data would also improve the interpretation and validation of the simulation result.

A better user interface that enables users to modify system variables would make the simulation more responsive to changes and eliminate the burden on users to learn another application just to input data. However, these improvements will require the transition from the Actor environment to a different modeling environment, as discussed below.

V.2.B. Limitations of the Actor environment

The use of an interpretative language environment like Actor, though easy to model, has some distinct disadvantages. Being interpretative, Actor uses a finite number of object pointers to keep track of the dynamic creation and termination of objects. Discrete event simulation requires the presence of large numbers of entities and events. In the case of an object-oriented system, this means a large number of objects. The "garbage collection" engine in Actor, responsible for freeing pointers to objects that no longer exit, must work harder in order to process larger numbers of objects, thus, consuming more processing time for internal system maintenance.

This object pointer limitation has two important implications to S3. First, the execution is very slow. For the example discussed in the previous chapter, it took approximately one hour real time to simulate one hour of operation on a personal computer equipped with an 80386 processor with a 33 MHz clock rate. Second, S3 ends up consuming all the available object pointers in Actor, thus bringing the application to a crash. This depends on the size of the model. In the example of the previous chapter, the model ran for four hours of simulation time before it crashed.

In addition to the pointer problem, Actor is inherently slow. To overcome these problems and to provide the speed of simulation more suitable for industrial applications, future versions of S3 should move to another language environment such as C/C++. Moving to such platforms may also make it possible to incorporate input and output facilities in the model rather than as external modules.

REFERENCES

Adams, Edward. L. 1984a. DESIM: A System For Designing And Simulating Hardwood Sawmill Systems. USDA Forest Service, General Technical Report NE-89.

Adams, Edward. L. 1984b. DSMIN user's Manual: A Procedure For Designing And Simulating Hardwood Sawmill Systems. USDA Forest Service, General Technical Report NE-94.

Adams, Edward. L. 1985. DESIM data manual: a procedure guide for developing equipment processing and down time data. USDA Forest Service, General Technical Report NE-102.

Adams, Edward. L. 1988. "Testing DESIM For Designing And Simulating The Operations Of Hardwood Mills". *Forest Product Journal*, 38(7/8): 41 - 45.

Araman, P. A. 1977. Use Of Computer Simulation In Design And Evaluating A Proposed Rough Mill For Furniture Parts. USDA Forest Service, Research Paper NE-361.

Aune, Jan E. 1973. The Application Of Simulation Technique In The Study Of Sawmill Productivity. Masters thesis. Faculty of Forestry, University of British Columbia, Vancouver, British Columbia.

Aune, Jan E. 1974. "System Simulation - A Technique For Sawmill Productivity Analysis And Designs". *Forestry Chronicle*, 50(2): 66 - 69.

Aune, Jan E. and E. L. Lefebvre, 1975. Small-log Sawmill Systems In Western Canada. Forintek Canada Corporation, Western laboratory Information Report VP-X-141.

Briggs, L. J., 1978. Sawmill Design Analysis Using Computer Simulation. Masters thesis. College of Forest Resources, University of Washington, Seattle, Washington.

Cox, B. J. 1986. *Object-oriented programming: an evolutionary approach*. Addison-Wesley, New York, New York.

Goldberg, Adele and David Robson, 1989. *Smalltalk-80: The Language*. Addison-Wesley, Reading, Massachusetts.

Hall, R. J. and D. B. Jewett. 1988. "A simulator for the design, analysis, and management of sawmills". *A paper presented at: Forest Products Research Society 42nd annual meeting*, Quebec city, Quebec.

Hallock, Hiram and David W Lewis, 1971. Increase Softwood Dimension Yield From Small Logs - Best Opening Face. USDA Forest Service, Research paper FPL-166.

Holmes, Stewart, 1976. "Introduction to operations research as applied in forest products industries". *Forest Products Journal*, 26(1): 17 - 22.

Kempthorne, K. H. 1978. "Whole mill simulation of small log sawmills with head sawyers". *IEEE, Miami Beach, Florida, Proceedings of winter simulation conferences*, 2: 684 - 692.

Kline, D. E. and J. K. Weidenbeck, 1989. "Design and evaluation of hardwood processing facilities using simulation/animation". *American Society of Agricultural Engineers, a presentation at the 1989 International Winter Meeting of ASAE*, New Orleans, Louisiana.

Knuth, Donald E. 1981. *The Art of Computer Programming*, Volume 2, 2nd Ed. Addison-Wesley, Reading, Massachusetts.

Law, Averill M. 1990a. "Simulation software for manufacturing applications: the next few years". *Industrial Engineering*, 22(6): 14 - 15.

Law, Averill M. 1990b. "Simulation software for manufacturing applications: part 2". *Industrial Engineering*, 22(7): 18 - 74.

Law, Averill M. and W. David Kelton, 1991. *Simulation Modeling and Analysis*, 2nd Edition. McGraw-Hill, New York, New York.

Martin, J. 1971. General purpose sawmill simulator for hardwood sawmills in the Northeast. Masters thesis. School of Forest Resources, Pennsylvania State University, University Park, Pennsylvania.

McAdoo, J. C. 1969. "Computer simulation of small-log mill processing". *Forest Products Journal*, 19(4): 34 - 35.

Meimban, Roger J., 1991. Simulation of hardwood sawmilling operations (Sawmilling). Doctorate dissertation. Pennsylvania State University. University Park, Pennsylvania.

Microsoft Corporation, 1990. *Microsoft Disk Operating System Users Guide, Version 5.0.* Microsoft Corporation, Redmond, Washington.

Microsoft Corporation, 1991. *Microsoft Windows Users Guide, Version 3.1.* Microsoft Corporation, Redmond, Washington.

Microsoft Corporation, 1992. *Q+E for Microsoft Excel Users Guide, Version 4.0 for Windows series.* Miscorsoft Corporation, Redmond, Washington.

Orbay, Lazlo. 1984. Computer simulation of overseas product manufacturing potentials of a redesigned multipass headrig mill in coastal British Columbia. Doctorate dissertation. Faculty of Forestry, University of British Columbia, Vancouver, British Columbia.

Pegden, C. 1989. *Introduction to SIMAN.* System Modeling Corporation, Swickley, Pennsylvania.

Penick, E. B. Jr., 1969. "Monte Carlo simulation for production control". *Forest Products Journal*, 19(5): 10.

Pritsker, A. A. B. 1974. *The GASP IV Simulation Language.* John Wiley & Sons. New York, New York.

Pritsker, A. A. B. 1986. *Introduction to Simulation and SLAM II.* 3rd Ed. John Wiley & Sons, New York, New York.

Randhawa, Sabah U., Guangchao Zhang, Charles C. Brunner, and James W. Funck, 1989. "S3: A Microcomputer-based Simulation Model For Sawmill Design And Evaluation". *In: Proceedings of International Symposium On Computer Applications In Design, Simulation, and Analysis, International Society For Mini and Microcomputers*, Reno, Nevada, 290 - 293.

Richard, C. A. 1979. Predicting the potential effect of the use of controlled log diameter sequencing at the warm spring forest products pine sawmill. Doctorate dissertation. College of Forest Resources, University of Washington, Seattle, Washington.

Savsar, M. 1982. A simulation model for sawmill operations. Masters thesis. School of Forest Resources, Pennsylvania State University, University Park, Pennsylvania.

Savsar, M. and P. C. Kersavage, 1982. "Mathematical model for determining the quantity of material produced in sawmilling". *Forest Products Journal*, 32(11-12): 35 - 38.

Schribner, T. 1974. *Simulation Using GPSS*. John Wiley & Sons, New York, New York.

The Whitewater Group, 1991a. *Actor Programming, Version 4.0*. The Whitewater Group, Incorporated, Evanston, Illinois.

The Whitewater Group, 1991b. *Actor SQL Reference, Version 4.0*. The Whitewater Group, Incorporated, Evanston, Illinois.

Tochigi, Toshiro, Reiichi Amemiya, Chiaki Tadokoro, Hikaru Hashimoto. 1988a. "Improving Production System Of Timber- processing Plants I: Development Of The Computer-simulation Procedure To Design The Production Line". *Journal of the Japan Wood Research Society*, 34(4): 320 - 325.

Tochigi, Toshiro, Reiichi Amemiya, Chiaki Tadokoro, Hikaru Hashimoto. 1988b. "Improving Production System Of Timber- processing Plants II: Appropriate Positioning Of Processing Machine In Sawmills". *Journal of the Japan Wood Research Society*, 34(4): 325 - 332.

Wagner, Francis G. Jr. 1982. Simulation modeling - a useful tool for sawmill decision making. Doctorate dissertation, Mississippi State University, Mississippi State, Miss.

Wagner, Francis G. Jr., R. D. Seale and Fred W. Taylor, 1989. "MICRO-MSUSP: An Interactive Computer Simulation Program For Modeling Southern Pine Sawmills". *In: Executive summary of 44th Forest Products Research Society Annual Conference*, Reno, Nevada.

Wagner, Francis G. Jr. and Fred W. Taylor, 1975. "Simulated Sawing With A Chipping Headrig". *Forest Products Journal*, 25(10): 24 - 28.

Wagner, Francis G. Jr. and Fred W. Taylor, 1983a. "SPSM - A Southern Pine Sawmill Model". *Forest Products Journal*, 33(4): 37 - 43.

Williston, Edward. M. 1990. *Lumber Manufacturing: The Design And Operation Of Sawmills And Planer Mills*, 3rd Ed. Miller Freeman, San Francisco, California.