Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

AUTOMATIC MODEL INITIALIZATION FOR REAL-TIME DECISION SUPPORT

Lotfi K. Gaafar
Javeed Shaik

Department of Industrial Engineering
Clemson University
Clemson, SC 29634-0920

ABSTRACT

The research presented in this paper reports on a shop-floor simulation interface (SFSI) which utilizes an integrated bar code-based data acquisition system to improve the real-time performance of an existing simulation-based decision support system and expand its capabilities to shop-floor monitoring and prediction. SFSI utilizes the data acquisition system to detect any shop-floor problem, and to update and initialize simulation models to the current shop-floor status. SFSI can process the simulation model, determine its initial conditions' requirements, retrieve the required information from the data acquisition system, and update the simulation model. This allows the simulation to start with exact current system conditions eliminating the need for time consuming warm-up periods, thus improving the system's accuracy and real-time performance.

1 INTRODUCTION

Discrete event simulation has been used extensively to address the manufacturing system design problem. It provides a flexible tool capable of dealing with all the detail design decisions that must be made before systems can become operational (Paul and Flanagan 1991). This is achieved by experimenting with a model of the system to analyze its behavior under different scenarios. In most cases, the simulation model is discarded after the modeled system is implemented. Considering the resources required to develop and validate a simulation model, and the fact that the same model, with minor changes, could continue to represent the modeled system, a case can be made for continuing to use the same simulation model throughout the operational life of the modeled system. Using simulation for operational decisions becomes more appealing when we consider the complexity of the manufacturing environment and the need for quantitative analysis to evaluate the impact of alternative decisions (Schmidt 1984). However, for a simulation system to support operational decisions, it must provide an easy way of modifying the simulation model to match the continuously changing system conditions, and must present results to the decision maker in a timely fashion.

This paper outlines a system structure that addresses both issues. In this approach, the simulation system is interfaced to a shop-floor data acquisition system (DAS) to detect shop-floor problems by continuously monitoring the collected data. When a problem is detected, a simulation model of the shop-floor is updated and initialized, using the same data, to match the current shop-floor conditions. Simulation is then executed to predict the effect of the detected problem on the overall production plan. Alternatively, a problem could be specified manually by a user through a user interface, in which case the model is updated using the user specifications and initialized using actual shop-floor data from the DAS.

As manufacturing systems are usually non-terminating, one of the problems that slows down the simulation response is the need to eliminate the initial bias introduced by starting the system in an empty/idle state (Pegden et al. 1990). However, in the outlined system, this problem is eliminated since simulation execution starts with the current shop-floor conditions. Eliminating the initial bias speeds up the model execution, thus providing critical support for real-time simulation.

Current applications of real-time simulation are focused mainly on shop-floor control. Mannivannan and Banks (1991) developed a generalized framework for controlling a manufacturing cell. This framework utilizes a temporal knowledge-base to synchronize events and their times of occurrences in both an actual manufacturing cell and its simulation model. A dynamic knowledge-base, implemented using frame structures, is used for storing the results of the

simulation. An event/time synchronization module ensures that the events and event times used in simulating alternative control scenarios are in agreement with the corresponding values in the manufacturing cell. The system is built using SIMAN (Pegden et al. 1990) on a Sun workstation with the dynamic knowledge-base implemented in common Lisp.

Another application of the integration of simulation software to a shop-floor control system is discussed in Johnson et al. (1992). In their work, the simulation model of the manufacturing system is interfaced to the shop-floor control system allowing the control logic to be tested for flaws, before it is implemented on the shop-floor, to eliminate false starts. This type of integration has also been broadly discussed by other authors including Erickson et al. (1990), Harmonosky and Barrick (1988), and Harmonosky (1990), with emphasis on the benefits of a real-time decision support system and the implementation problems of such a system.

As stated before, the system presented in this paper goes beyond shop-floor control to achieve the goal of continuously monitoring the shop-floor status to automatically detect any potential problems. Once a problem is detected, the simulation model is automatically updated to the current shop-floor conditions, simulation is executed, and results are presented to the user with a warning message if current production plans cannot be achieved. The subsequent sections discuss the system structure and explain its utility using a case example.

2 SYSTEM STRUCTURE

The system presented in this paper, SFSI, is based on RTST (real time simulation tool [Gaafar and Cochran 1989 and Cochran and Gaafar 1990]). RTST is a tool which allows a user with no simulation experience to interactively modify and experiment with an existing simulation model within limits imposed by the expert model developer to insure model validity. SFSI adds a DAS interface to RTST to allow shop-floor events to be automatically monitored and to provide the required information to start simulation using real-time shop-floor data. Therefore, with the DAS interface, simulation can be triggered either manually, through the user interface in RTST, or automatically, as a result of continuously analyzing the data collected by the DAS. The overall system structure and functional logic of SFSI are shown in Figure 1. The following discussion explains each module in the system and the overall logic of SFSI.

The Data Acquisition System (DAS) continuously collects data related to the man, machine, and material components of the manufacturing cell. Specifically, the data collected consist of time/attendance, machine status, part location, queue levels, transporter status, and supervisory information. The DAS is bar code-based with manual input capabilities (Gaafar 1989). Data collected from the DAS is stored in the Dynamic Database. As a result, this database stores the current shop-floor data that include queue levels, machine and transporter status, and transporter location. This data is continuously compared to pre-defined performance levels (stored in the Static Database) to detect any alarming event (defined as a system parameter exceeding one of the pre-defined performance levels). In case of an alarm, the Dynamic Database provides the current shop-floor conditions to the Model Initializer which updates and initializes the simulation model.

The Static Database includes information regarding the pre-defined shop-floor performance levels and other production parameters. Pre-defined performance levels include the expected queue levels on each machine and different resource utilization levels. Production parameters include production mix, the inter-arrival times of different parts, the processing times of the parts on individual machines, and the current production schedule. The static database can be updated manually using the menu-driven user interface in RTST.

The Simulation Model Database stores SIMAN simulation models of different parts of the shop-floor. SIMAN requires two files, the model file and the experiment file, to complete a single simulation model. The model file typically describes the static and the dynamic characteristics of the model, while the experimental file defines the experimental conditions under which the model is to be executed (Sturrock and Pegden 1990).

The major contribution of this work comes from the ability of SFSI to automatically initialize any simulation model, written in SIMAN, to reflect the current shop-floor conditions. The Model Initializer performs this function by updating different fields in the experimental file of the simulation model. Presently SFSI has the capability to initialize queue levels, machine status, transporter locations, and transporter status, and incorporate any anticipated machine breakdowns. Fields in the experimental file are initialized using information obtained from the Dynamic Database.

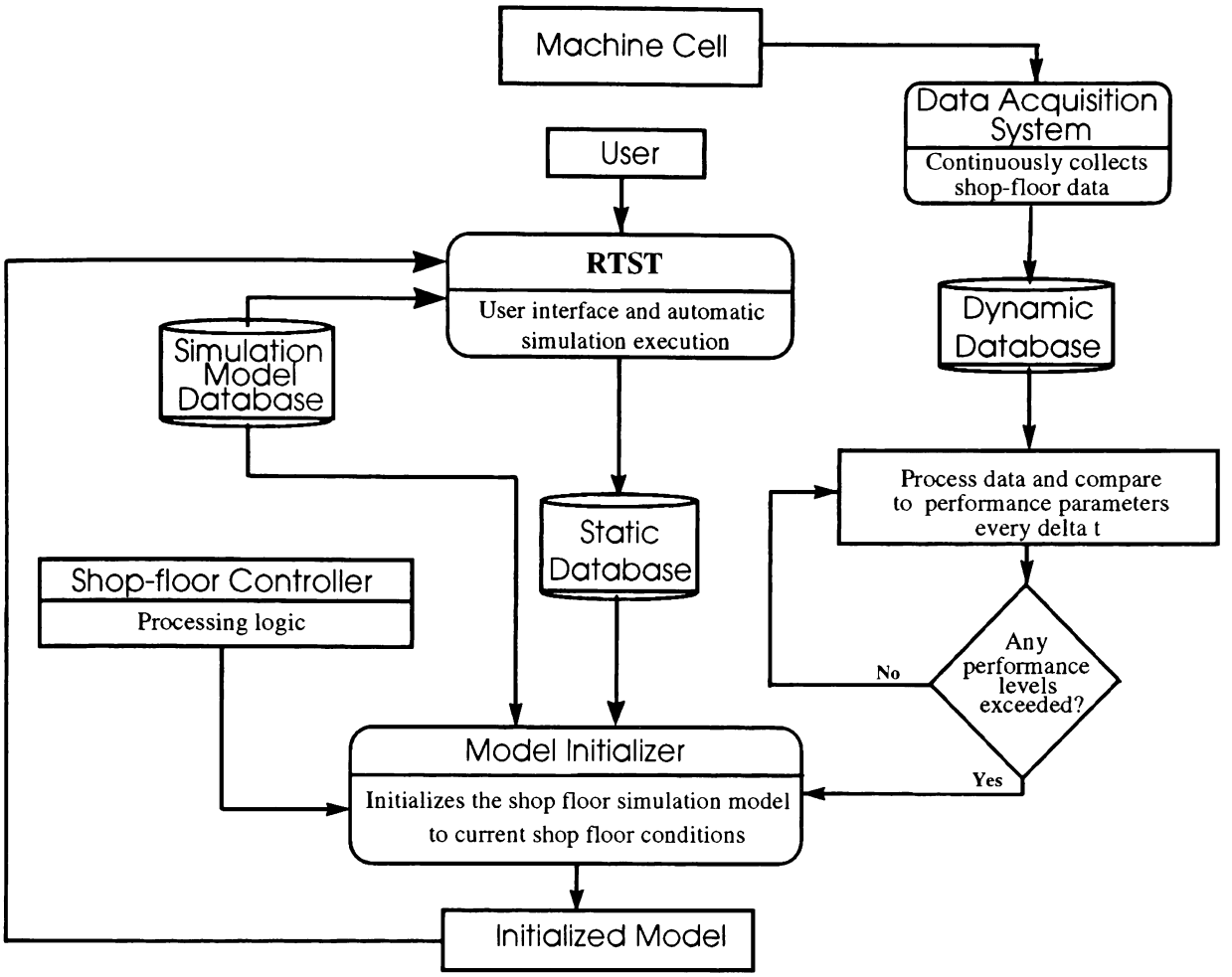

Figure 1: SFSI System Structure and Functional Logic

Work is being carried out to expand the capabilities of the Model Initializer to incorporate additional data including job priorities, job routing sequences, and part arrival rates.

Model initialization is triggered in one of two cases:

1. the shop-floor data on queue levels, machine status, or resource utilization exceed the pre-defined limits set in the static database, or
2. a manual change is introduced by the user to experiment with different production scenarios.

The Shop-floor Controller contains the routing sequence of all parts and the overall processing logic. By interfacing the Model Initializer to the Shop-floor Controller, control decisions (such as a change in job priorities to accommodate a hot job or stopping a machine for a scheduled maintenance) can be automatically detected and incorporated into the simulation model. This interface has not been implemented yet, but is planned for the next phase in this research. The following section will illustrate some of the features of SFSI through an example that has been shortened for space considerations.

3 CASE EXAMPLE

In this example, we will consider a manufacturing cell consisting of two manufacturing stations. Station number 1 runs two identical machines in parallel and station number 2 runs one machine. Two products are manufactured in this cell. Each product has its own production sequence. The cell layout and part visitation sequences are shown in Figure 2. Parts are routed between stations via a fork lift truck.

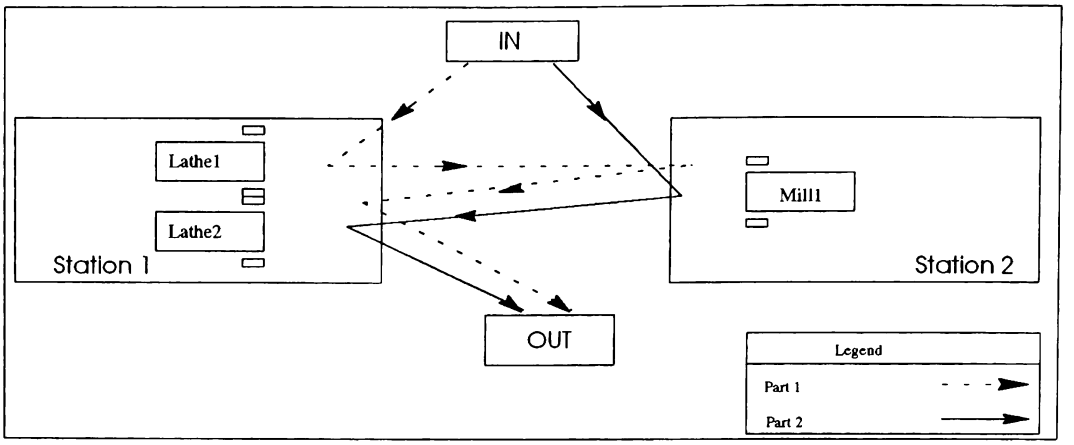

Figure 2: The Cell Layout and Part Visitation Sequences

In the following discussion, we will assume that a user wants to investigate the overall system performance after scheduling an hour of maintenance on Lathe1 two hours into the shift. The SIMAN simulation model for the system above consists of a model file and an experimental file. The experimental file is shown in Figure 3. Because of space limitations, and since the proposed changes will only affect the experimental file, the model file is not shown.

Starting from the main menu in RTST (not shown), the user may select one of five options: generate a new model, modify a model, execute simulation, display output, or quit. In our example, the user chooses the option to modify a model. This brings up the modification menu (also not shown) which allows the user to select a model from a list of available models and select a modification option. In our example, the user will choose the option to modify resources. This will bring up the resource modification menu (Figure 4) which is used to modify the parameters of any resource.

As shown in Figure 4, the resource modification menu allows the user to choose any resource and specify a scheduled break or alter its capacity. In our example, the user specifies the start time and the duration of the scheduled maintenance (120 and 60 respectively). The supervisor then quits this menu and selects the option to execute simulation from the main menu.

The changes made by the shop-floor supervisor are conveyed to the Model Initializer module of SFSI. The Model Initializer runs the appropriate subroutines to incorporate the changes into the experiment file. In this example the SIMAN element 'SCHEDULES' is added to the experimental file to model the scheduled maintenance. At the same time, the Model Initializer scans the Dynamic Database for the current shop-floor conditions. The Dynamic Database shows two jobs, of type 2, in Lathe2's queue (L2Q). The Model Initializer incorporates this information into the experiment file using the SIMAN element 'ARRIVALS' which is used to pre-load queues.

```
   BEGIN;
1  PROJECT,            Robot Cell, Clemson Univ, 1/1/1993;
2 ATTRIBUTES:          1,time_in:2,op_time;
3 VARIABLES:           1,inventory1,100:
                       2,inventory2,250;
4 QUEUES:              1,INQ:2,L1Q:
                       3,L2Q:4,MQ:5,OUTQ;
5 RESOURCES:           1,Lathe1,1:2,Lathe2,1:
                       3,Mill1,1;
6 STATIONS:            1,IN:2,Lathe:3,Mill:4,OUT;
7 TRANSPORTERS:        Truck,,1,60 - 0.0 - 0.0 - 1.0, 1-
                       Active - ZONE(1);
8 SEQUENCES:           1, IN&1,optime=20&2,op_time=25&
                       1,op_time=15&OUT:
                       2, IN&2,op_time=30&1,op_time=25&
                       OUT;
9 DISTANCES:           1, 1-4, 120, 200, 140/ 50, 125/ 125;
10 TALLIES:            1,Time in IN1 Queue:
                       2,Time in IN2 Queue;
11 DSTATS:             nq(1),Number in OUT1 Queue:
                       nq(2),Number in OUT2 Queue;
12 REPLICATE,          ,,7200;
   END;
```

Figure 3: The Experimental File for the System in Figure 2

The 'ARRIVALS' element requires other parameters including the station number where the part is currently residing, the routing sequence followed by the part, and

the current index within the part sequence. SFSI updates these parameters to 1, 2, and 3 respectively. The entity attributes, time_in (time when the entity entered the system) and op_time (time required to process the entity on Lathe2) , are also updated as part of the 'ARRIVALS' element. Figure 5 displays a small portion of the experimental file where the above mentioned changes have been incorporated. These changes have been underlined in Figure 5 for clarity.

As outlined in Figure 1, the initialized file is transferred to RTST which executes the simulation for the specified time interval (7200 time units) and displays the results. The RTST output report (Figure 6) shows that the system is unbalanced because of the over utilization of Lathe1. The utilization of Lathe1 is 100% which exceeds the pre-defined utilization limits (90%) in the Static Database, designed to account for emergencies. The report also shows that the production schedule will not be completed. RTST has the capability to generate graphical output on any of the different statistics under consideration. The user can visualize the extent of over-utilization by selecting the appropriate graph from the graphics display menu (Cochran and Gaafar 1990).

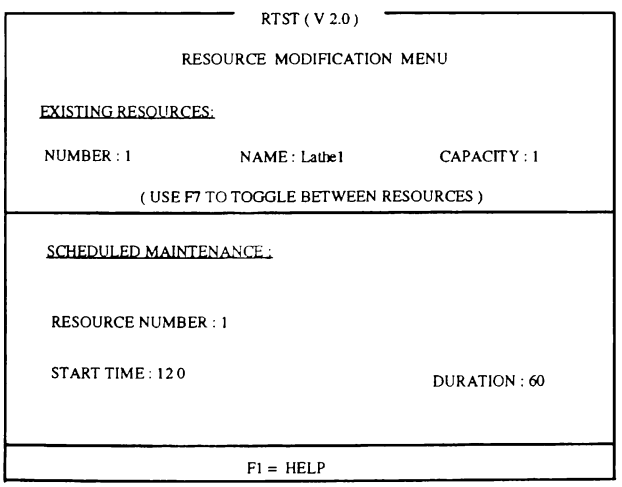

Figure 4: The Resource Modification Menu

The output clearly indicates to the user that scheduling Lathe1 two hours from the start of the shift for preventive maintenance is not a feasible option. Following the same steps, the user can now analyze other options of scheduling maintenance on Lathe1 (such as shortening the maintenance time, or rerouting parts).

In this example, SFSI has helped the user to 'look-ahead' and analyze the possible results of a decision on the cell productivity while automatically incorporating information not supplied by the user. In this respect, SFSI works effectively as a decision support system for the shop-floor personnel to analyze routine shop-floor problems in real-time. While this example showed an SFSI session based on a user triggered event, the system will follow similar logic if an event was automatically detected when a performance level is exceeded. For example, if the Dynamic Database shows 6 parts in the L2Q while the limit in the Static Database is only 5, simulation will automatically be executed after SFSI updates the model to the current shop-floor status.

```
 BEGIN;
1 PROJECT,          Robot Cell, Clemson Univ,1/1/1993;
  |                                  |
  |                                  |
5 RESOURCES:        1,Lathe1,SCHED(1): 2,Lathe2,1:
                    3,Mill1,1;
6 ARRIVALS:         1,queue(2),,1,30,25,1,2,3:
                    2,queue(2),,1,54,25,1,2,3:
7 SCHEDULES:        1,1*120,0*60,1:
|                                  |
|                                  |
 END;
```

Figure 5: Part of the Experimental File After Changes

4 SYSTEM IMPLEMENTATION

As mentioned earlier, SFSI is developed using RTST as a platform. The data extraction and model initialization capabilities were added to the RTST program by incorporating additional routines (using the C programming language) in two separate modules.

Current implementation efforts are focused on the automatic interfaces between the DAS and the Dynamic Database, and between the Model Initializer and the Shop-Floor Controller, which are not yet fully developed.

SFSI is implemented on a personal computer (PC) platform which was chosen because of the PC availability at the shop-floor level and its great integration potential. Even though SFSI will run on any PC configuration, an 80386 microprocessor with clock speed of 20 MHz or more, and a math co-processor are highly recommended. The SFSI session described in the previous section required approximately 6 minutes on the PC/386 platform.

```
*************************
RTST OUTPUT REPORT
*************************
WARNINGS!!

PERFORMANCE LIMITS ARE EXCEEDED.
RESOURCE # 1 (Lathe1) IS OVER
UTILIZED (100%)

PRODUCTION SCHEDULE FOR PART #1 IS
NOT SATISFIED (90 %)

THE FOLLOWING GRAPHS ARE AVAILABLE :

TYPE                              No.
Queues                            1 - 5
Resources Utilization:
Lathe1                            6
Lathe2                            7
Mill1                             8
Transporters:
Truck                             9
Enter a graph number ( 0 to quit ) :

END RTST REPORT
```

Figure 6: Output Report from RTST

6 CONCLUSION

SFSI is a system that extends the useful life of a simulation model by supporting model reusability.

This is achieved by automatically updating the simulation model based on changes on the shop-floor, using data obtained from a DAS.

A byproduct of the DAS interface is the ability to start simulation using current system conditions which eliminates the need for long warm-up periods and supports real-time execution. SFSI continuously monitors the shop-floor status to automatically detect any potential problems. When a problem is detected, the simulation model is automatically updated to the current shop-floor conditions, simulation is executed, and results are presented to the user with a warning message if current plans cannot be achieved.

The database interfaces in SFSI are still under development, while the Model Initializer is fully functional. While 6 minutes could be considered a reasonable response time for the presented case example (which involved 15 replications), it only represents one scenario or decision alternative. The user might have to consider many other alternatives, in which case a faster response will be required. Considering that more powerful PC platforms (e.g., a PC/486 with a clock speed of 66 MHz) are available at reasonable prices, the system response can be greatly improved at limited expenses. Future research will focus on the database interfaces and adding an automatic model selection capability to SFSI. In this case, SFSI would be able to automatically select the appropriate simulation model from the Simulation Model Database based on the triggering event.

REFERENCES

Cochran, J. and Gaafar, L. 1990. "An integrated Shop-Floor Simulation-Based Decision Support Tool for Shop-Floor Environments." *Proceedings of the 1990 Computer Simulation Western Multiconference*, pp. 81-86, San Diego, CA.

Erickson, C., Vandenberge, A., and Miles, T. 1989. "Simulation, Animation, and Shop-Floor Control." *Proceedings of the 1990 Winter Simulation Conference*, pp. 649-653, San Diego, CA.

Gaafar, L. 1989. *An integrated Shop-Floor Simulation-Based Decision Support Tool,* Unpublished Masters Thesis, Arizona State University, Tempe.

Gaafar, L. and Cochran, J. 1989. "Developing a Real-Time Simulation Tool for Shop-Floor Decision Making." *Proceedings of the Summer Computer Simulation Conference*, pp. 79-85, Austin, TX.

Harmonosky, C. 1990. "Implementation Issues using Simulation for Real-Time Scheduling." *Proceedings of the 1990 Winter Simulation Conference,* pp. 595-598, San Diego, CA.

Harmonosky, C. and Barrick, D. 1988. "Simulation in CIM Environment : Structure for Analysis and Real-Time Scheduling." *Proceedings of the 1988 Winter Simulation Conference,* pp. 704-711.

Johnson, E., Thompson, L., and Fontaine, R. 1992. "An Integrated Simulation and Shop-Floor Control System." *Manufacturing Review*, Vol. 5, No. 3, pp. 158-165.

Mannivannan, S. and Banks, J. 1991. "Real-Time Control of a Manufacturing Cell Using Knowledge-

Based Simulation." *Proceedings of the 1991 Winter Simulation Conference*, pp. 251-260, Phoenix, AZ.

Paul, R. and Flanagan, M. 1991. "On-line Simulation for Real-Time Scheduling of Manufacturing Systems." *Industrial Engineering*, Vol. 23, No. 12, pp. 37-40.

Pegden, C., Shannon, R., and Sadowski, R. 1990. *Introduction to Simulation Using SIMAN*, McGraw-Hill, Inc., Princeton Road, Hightstown, New Jersey 08520.

Schmidt, J. 1984. "Introduction to Simulation" *Proceedings of the 1984 Winter Simulation Conference*, pp. 65-73.

Sturrock, D. and Pegden, D. 1990. "Introduction to SIMAN." *Proceedings of the 1990 Winter Simulation Conference*, pp. 109-114, San Diego, CA.

AUTHOR BIOGRAPHIES

LOTFI K. GAAFAR is an Assistant Professor of Industrial Engineering at Clemson University. He received the MSIE and Ph.D. degrees form Arizona State University. His current interests are in manufacturing modeling and simulation. He is a senior member of SCS, IIE, and SME.

JAVEED SHAIK is a graduate student in the Department of Industrial Engineering at Clemson University. He is researching computer simulation in real-time control. He received the BS degree in Production and Industrial Engineering from R. V. College of Engineering, India. He is a student member of IIE and ASQC.