Improving Verification Skills for a Discrete-Event Simulation Model

Dr. Sima Parisay, California State Polytechnic University, Pomona

Dr. Sima Parisay is an Emeritus Professor from the Industrial and Manufacturing Engineering Department at California State Polytechnic University, Pomona, CA. In recent years, she has been a part-time lecturer in the Industrial and System Engineering Department at University of Southern California, Los Angeles, CA. She has 28 years of teaching experience, including creating pedagogical tools to improve the learning process and critical thinking in engineering topics. She is dedicated to not only teaching concepts to students, but also ensuring they acquire important skills useful for their future career endeavors. Outside of the classroom, she has been active, in various capacities, with the Pacific Southwest Section (PSW) of ASEE, American Society for Engineering Education.

She received a B.S. in Industrial Engineering from Sharif University in Iran; M.S. in Production Engineering from Aston University in U.K.; and Ph.D. from Industrial and Systems Engineering at the University of Southern California, in U.S.A.

To reach her, please email: parisay@usc.edu.

Improving Verification Skills for a Discrete-Event Simulation Model

Abstract

For over two decades of teaching discrete-event simulation courses, a consistent problem has been noted: most students lack adequate skills needed to verify their simulation models. Compounding this issue is the confusion many students have regarding the distinction between debugging and verification. Most often, professional versions of simulation software, adept at running models and producing output, overlook logical errors. This can lead students to mistakenly believe their models are flawless. Additionally, some students rely solely on animation of their models for verification, which, though helpful, cannot provide the precise and dependable information necessary for a thorough verification process. Unfortunately, existing textbooks and literature often provide limited discussion on model verification, leaving students inadequately prepared.

This paper introduces a set of comprehensive guidelines designed to facilitate the systematic verification of simulation models. To illustrate the practical application of these guidelines, the paper showcases examples. By following these guidelines, students can refine their verification skills and achieve more reliable models, moreover develop critical thinking abilities throughout the process.

This paper also proposes a special-purpose flowchart called a "Logical Model." This model serves a valuable role throughout the simulation process, including: initial system study, model creation, semantic debugging, and verification. The discussion on additional applications of the Logical Model, such as analysis and design of experiments, will be presented in a later paper.

1. Introduction and literature review

Like any scientific study, a simulation project involves a defined sequence of steps. The process begins with establishing a clear problem statement and study goals, leading to actionable recommendations to support decision-making. Crucially, after model development, simulation studies require debugging, verification, and validation. However, some students struggle to distinguish between these terms.

Debugging entails meticulously examining a model for errors. This encompasses syntax issues as well as semantic or logical errors that can lead to misleading output. Fortunately, advanced simulation software like Arena and ProModel offer built-in features to detect syntax errors and provide informative error messages.

Distinguishing verification and validation can be challenging. Sargent and Balci [1] noted: "The definition of Verification and Validation have been inconsistent over time." They then provided clear definitions: "Verification is concerned with ensuring the programmed model behaves as the users believe it does," while "Validation is concerned with whether the model adequately represents a system for the model's purpose." These definitions align with those found in numerous resources, including Kelton et al. [2]. This paper adopts this definition for verification.

While existing literature explores verification and validation, the author finds it inadequate for educational purposes. Sargent and Balci [1] delve into the historical context of these concepts, while Whitner and Balci [3] offer general verification guidelines. Balci [4] expands on these for both verification and validation, but their scope is more suited to large-scale projects than introductory courses. Additionally, these pre-2000 works predate the advanced capabilities of modern simulation software. Although recent publications like [2], [5], [6], and [7] address verification, they often lack practical examples.

Below is a compiled list of suggested verification approaches based on these references:

- Entity tracing: Follow one or a few entities through the model, verifying their journey aligns with expectations and output reflects expected information.
- Data simplification: Replace some variable input data with fixed values to predict system behavior under controlled conditions.
- Model simplification: Run a simplified model. Examples of simplification include: a model while its resource does not have break times or failures, or its queue capacity is unlimited and there is no balking.
- Extreme scenario Testing: Test the model under extreme conditions, like significantly altered arrival times or service times, to identify potential weaknesses.
- Animation review: Observe the model's animation for visual confirmation of expected behavior and entity flow.
- Long-term run analysis: Run the model for an extended period, checking for excessive queue buildup or underutilized resources, which could indicate inefficiencies.
- Model decomposition: Develop the model incrementally by building and testing smaller sub-models before integrating them into the entire system.

Over 28 years of teaching discrete-event simulation to undergraduate and Master's students in Industrial Engineering have revealed a consistent challenge: applying the above verification approaches effectively. While valuable, these approaches can be time-consuming, and students may lack the experience or background knowledge to use them optimally. To maintain focus within this paper, the author will not delve into detailed examples of these challenges.

In response to these student difficulties, the author developed the “Logical Model,” a simple yet powerful tool. This model is essentially a flow diagram specifically designed for the simulation study process. To solidify its application, students are required to draw a Logical Model for every assignment, quiz, exam, and project.

The Logical Model, for its first application, serves a dual purpose. First purpose, it assists students in constructing their models by providing a clear visual framework. Second purpose, it facilitates semantic debugging after model creation by looking for logical inconsistencies. The second application of Logical Model is for initial system analysis. For the third application, it is used in conjunction with the author’s developed guidelines to help students through a systematic verification process.

Approximately 25 years ago, the author introduced the Logical Model concept and its application to simulation courses. Its initial purpose was to help undergraduate Industrial Engineering

students reduce modeling errors. In the last decade, implementation of this pedagogy in Master's-level simulation courses proved to be even more beneficial as students often lack an Industrial Engineering background. While a formal quantitative analysis has not been conducted of this recent experience, the pedagogy's effectiveness was evident. Encouraged by these positive results, the author continued to develop further applications for the Logical Model over the years, expanding its use into areas like verification, output analysis, and even design of experiments.

Appendix A showcases a common student mistake to illustrate the value of this new pedagogical tool. The example features a system deliberately designed to exhibit blow up (meaning it falls into an unsteady state with a queue that keeps growing, also known as pile up). Students often miss this potential issue when building a simulation model with software, leading to challenges during execution. These challenges are detailed within the Appendix.

While some students might attempt to identify blow up issues through animation features in the software, this method proves unreliable in this specific case. Animation, although a valuable tool for understanding certain system dynamics, cannot replace a systematic verification process.

This paper focuses on explaining the Logical Model designed for simulation studies. Subsequently, it presents a list of common performance measures used for verification. The main body provides four examples demonstrating this verification approach. Appendix A, delves deeper into the motivation for the Logical Model by presenting an initial system analysis before modeling. Appendices B to F, contain supporting documentation for the model outputs and calculations used in the four examples in the main body.

A basic understanding of classical queuing theory and discrete-event simulation concepts is assumed for this paper.

2. Logical Model

The Logical Model concept was introduced in a discrete-event simulation course in spring 1999. Its effectiveness in student's performance was tested in subsequent years. This model acts as a specialized flowchart, capturing all problem statement details and showcasing the entity flow. Importantly, the conventions used are independent of the modeling software, promoting a focus on "simulation science" rather than software-specific practices. Additionally, the model strives for simplicity, facilitating communication with non-experts in the field. Notably, the Logical Model has proven effective in reducing students' errors related to semantic errors.

The Logical Model serves multiple purposes. It is employed for initial system analysis, even before creating a simulation model (detailed in Appendix A). This initial analysis is crucial in projects where students make frequent assumptions without grasping potential consequences. Furthermore, the Logical Model aids in estimating performance measures and some data points. Such estimated values will assist with further verification.

The Logical Model convention uses boxes to represent groups of related information. Figure 1 illustrates a sample model. Each box focuses on a specific aspect. For instance, the "creating entities" box would contain details like the arrival point name (Entrance), entity name (Part), and

inter-arrival time distribution (Exponential with a mean of 6 minutes). Similarly, the "process" box would detail the process name (Station A), resources used (Machine), number of resources needed ([1,..) to process one entity, total number of parallel (identical) resources available (..3]), and service time distribution (Uniform with a minimum of 10 and a maximum of 19 minutes).

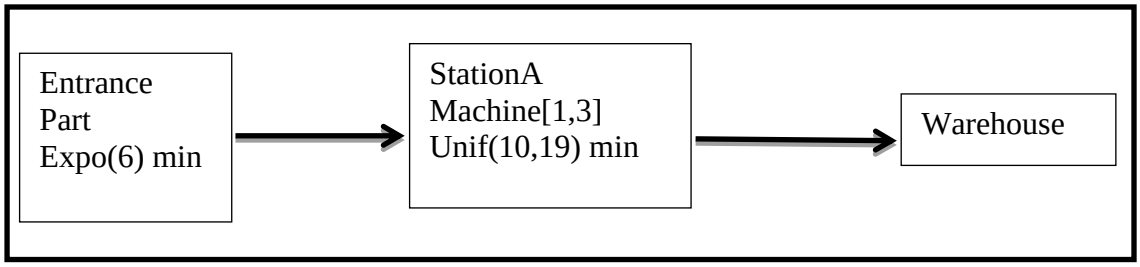


Figure 1: Sample Logical Model

In summary, the Logical Model, as presented and utilized in this paper, offers several key advantages: Facilitates initial system analysis, reduces errors in model creation, and opens the door for systematic verification.

3. Common verification metrics

This section outlines the key performance measures (PMs) and data points commonly used for model verification. Throughout this paper, these will be referred to as "verification metrics."

1. Utilization of resources: This metric indicates how busy a resource is compared to its available time. This is a PM.
2. Total average Value-Added (VA) time for each entity: This represents the average time spent by an entity undergoing processes. This is a PM.
3. Total average Non-Value-Added (NVA) time for each entity: This captures the average time an entity spends in activities that do not add value, such as rework time. This is a PM.
4. Total average transfer time for each entity: This metric reflects the average time entities spend moving between locations within the system. This is a PM.
5. Probabilities in branching: These probabilities determine the likelihood of entities taking different paths within the model. This is a data point.
6. Number of entities entered during simulation: This represents the total number of entities entering the system during the simulation. This is a data point.
7. Balk rate: This metric applies only when queue capacity is limited. It reflects the rate at which entities are balking (leaving the system) due to a full queue. This is a PM.

Four examples are presented to demonstrate how the above verification metrics are used for verification. Example 1 showcases application of verification metrics 1, 2, 3, 5, and 6. Example 2 focuses on verification of utilization (metrics 1), highlighting different calculation approaches. Example 3 demonstrates verification of probability branching. And finally, Example 4 emphasizes critical thinking when insufficient information exists for precise estimation. It showcases application of upper or lower bounds for verification and also applies the balk rate

metric. Additionally, it explores verification of average waiting time and average queue length using queuing theory principles.

Using a Logical Model, we can estimate the value of a required verification metrics (performance measures or data points). After running the simulation model, we obtain the values of these metrics from the output. Then we calculate the percentage of difference between the estimated and obtained values. This percentage of difference should be less than a subjectively selected threshold for being verified.

The value of this threshold can be decided based on many factors. Below are the author's suggested values based on teaching experiences:

Suggested threshold	Model specification
5%	Complex models with short run time, for example end of term projects
2%	Simple models with long run time, for example class level models

Law [8] uses two examples, Example 8 and 9, where he considers percent of differences to be between 1% and 11% for similar situations.

A percentage difference exceeding the predetermined threshold should not be viewed as definitive proof of an error. However, it does serve as a red flag, prompting a closer look at that specific section of the simulation model. This re-examination can help identify and rectify potential discrepancies between the model and the intended system behavior.

4. Example 1: Verifying multiple metrics in a parts processing system

This example serves as a practical introduction to the verification approach outlined in Section 3. It guides users, whether in a lecture setting or for self-study, through the process of verifying most of the metrics mentioned in that section.

Problem statement: Subsystem for parts processing

We are analyzing a subsystem for processing parts. Parts arrive individually at Station L with an average inter-arrival time of six minutes, following an exponential distribution (Expo(6)). This station has three identical (parallel) Machines dedicated to processing the Parts. Each Machine at Station L processes each Part with a processing time drawn from a triangular distribution, with a minimum of 10 minutes, a mode of 13 minutes, and a maximum of 19 minutes (Tria(10, 13, 19)).

After processing at Station L, 20% of the Parts are directed to Storage. The remaining 80% of the Parts proceed to Station M, where a single Operator performs another process. The processing time per Part at Station M is uniformly distributed between 3 and 10 minutes (Unif(3, 10)). Finally, all parts are sent to the same Storage location.

A constant travel time of five minutes is assumed for movement between stations, stations and Storage, and the entrance to Station L.

a) Create a Logical model
b) Simulate this system for 40,000 unit of time and 10 replications
c) Verify the model as much as possible; consider a 2% threshold for verification

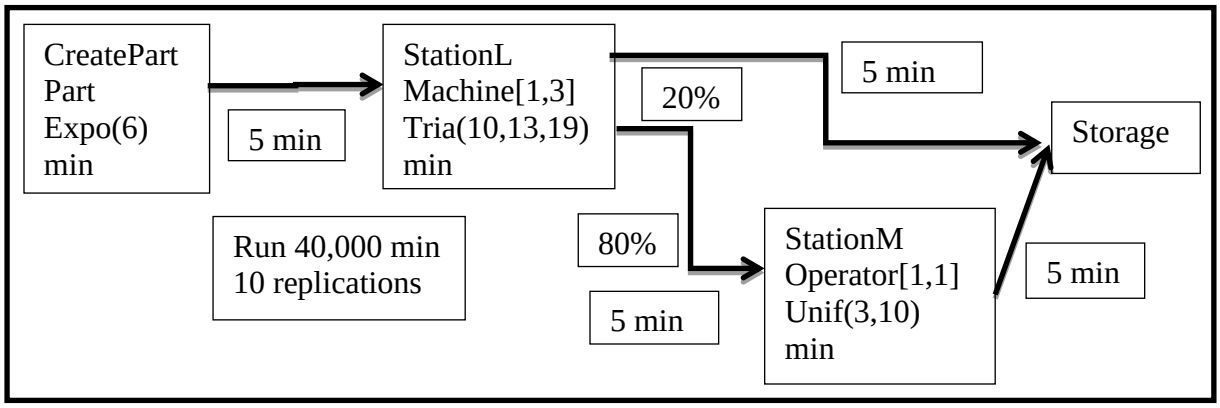


Figure 2: Logical Model of Example 1

Below is the calculation of required 'Estimated Values' using the Logical Model:

- Estimate utilization of each Machine at Station L: Parts arrive at Station L every 6 minutes on average (exponential distribution, Expo(6)). The average service time is 14 minutes (triangular distribution average= (10+13+19)/ 3). Utilization can be calculated as average service time divided by average inter-arrival time. Therefore, the estimated total utilization for Station L is 2.33 (14 minutes / 6 minutes). With three Machines, each Machine's utilization would be 0.778 (2.33/ 3).

  Alternatively, utilization can be calculated as follows. With an average inter-arrival time of Parts as 6 minutes, then the rate of arrival is 1/6 parts/min. With an average service time of 14 minutes, then the rate of service is 1/14 parts/min. Utilization can be calculated as arrival rate divided by service rate. Therefore, the estimated total utilization is 2.33 ((1/6)/ (1/14)). This value matches the previous calculation, as expected.

- Estimate utilization of Operator at Station M: Consider a subjective study period of one hour for this calculation. On average, 10 Parts arrive at the system per hour (inter-arrival time is 6 minutes). Since 80% go to Station M after Station L, approximately 8 Parts require service in each hour (10 * 0.8) at Station M. The average service time per Part at Station M is 6.5 minutes (uniform distribution average= (3+10)/2). This translates to a total of 52 minutes needed to serve all Parts at Station M every hour (8 parts * 6.5 minutes/part= 52 minutes). The utilization can be calculated as average total service time needed in one hour divided by average available time of resources in one hour. Therefore, the estimated utilization of the Operator at Station M is 0.867 (52 minutes / 60 minutes).

- Estimate travel time (Non-Value-Adding Time): About 80% of Parts will have a total travel time of 15 minutes (sum of travel times= 5+5+5) and 20% will have a total travel time of 10 minutes (sum of travel times= 5+5). Considering the weighted average of travel times, the estimated average travel time is 14 minutes (0.8 * 15 minutes + 0.2 * 10 minutes= 14 minutes).

- Estimate total service time (Value-Adding Time): About 80% of Parts will have on average a total service time of 20.5 minutes (sum of average service times = (10+13+19)/ 3 + (3+10)/ 2). About 20% of Parts will have on average a total service time of 14 minutes (average of triangular distribution: (10+13+19)/3). Considering the weighted average of service times, the estimated average service time is 19.2 minutes (0.8 * 20.5 minutes + 0.2 * 14 minutes).

- Estimate number of arrivals during simulation: With a simulation length of 40,000 minutes and an average inter-arrival of 6 minutes, approximately 6666.7 Parts are expected to enter the system during the simulation (40,000 minutes / 6 minutes/Part = 6666.7 Parts).

- Estimate number of entities at Station M (using Branching): Approximately 6666.7 Parts enter the system during simulation. Based on the 80% branching probability to Station M, the estimated total Parts entering Station M is 5333.3 (6666.7 Parts * 0.8= 5333.3 Parts). This information helps verify the branching logic within the simulation model.

The output of the simulated model is in Appendix B. The required values for verification are highlighted in the appendix.

Table 1 summarizes the comparison between the estimated values and the values obtained from the statistical output of the Arena model. While exact matches are not expected due to inherent randomness in simulation, the table reveals a key point. All percentage differences in the last column fall below 2%. Therefore, it can be concluded that the Arena model is verified.

Table 1: Summary Table for Verification of Example 1

Data	Statistical output	Estimation	% difference
Average total VATime	19.2	19.2	0%
Average total NVATime	14	14	0%
Operator utilization, Station M	0.86	0.867	0.8%
Machine utilization, Station L	0.77	0.778	1.03%
Parts entered system during simulation	6608.5	6666.7	0.8%
Parts visited Station M (Operator)	5283.7	5333.3	0.9%

5. Example 2: Verification of utilization with different approaches

This example focuses on verifying the utilization of resources within the system depicted in Figure 3. It highlights two key points:

1. Prioritizing ease of calculation and communication: When selecting an estimation approach for utilization, the author emphasizes choosing a method that is both easy to calculate and easy to explain to others.
2. Leveraging the Logical Model: The author recommends creating a version of Logical Model with average service times to simplify the estimation process (Figure4).

For the sake of brevity, other verification aspects are not covered in this example. This example serves as a valuable teaching tool, suitable for both classroom lectures (second discussion point) and self-study assignments.

Problem statement: A multi-station system with mixed parts

The system under consideration receives parts of two types: GK and Q. Parts GK arrive with an inter-arrival time uniformly distributed between 5 and 7 minutes. Upon arrival, 20% are designated as part G and directed to Station L. The remaining 80% are designated as part K and proceed to Station M.

- Station L: This station has a single Machine that processes each part G with a triangularly distributed processing time (Tria(6, 9, 15)). After processing, all part Gs are sent to Station M.
- Station M: This station has a single Operator. It processes each part G with an exponentially distributed processing time (Expo(3)) and each part K with a triangularly distributed processing time (Tria(2, 3, 7)). Both part Gs and Ks are then routed to Station P.
- Part Q: Parts of type Q also enter the system with an inter-arrival time uniformly distributed between 8 and 12 minutes. They are directly sent to Station P.
- Station P: This station has a single Computer that processes both part Gs and Ks with a triangularly distributed processing time (Tria(1, 2, 3)). The Computer processes parts Q as well for a constant processing time of 4 minutes per part. Finally, all parts are sent to storage.

All times are in minutes.

a) Create a Logical model
b) Simulate this system for 30,000 unit of time and 10 replications
c) Verify the model using information on utilizations; consider a 2% threshold for verification

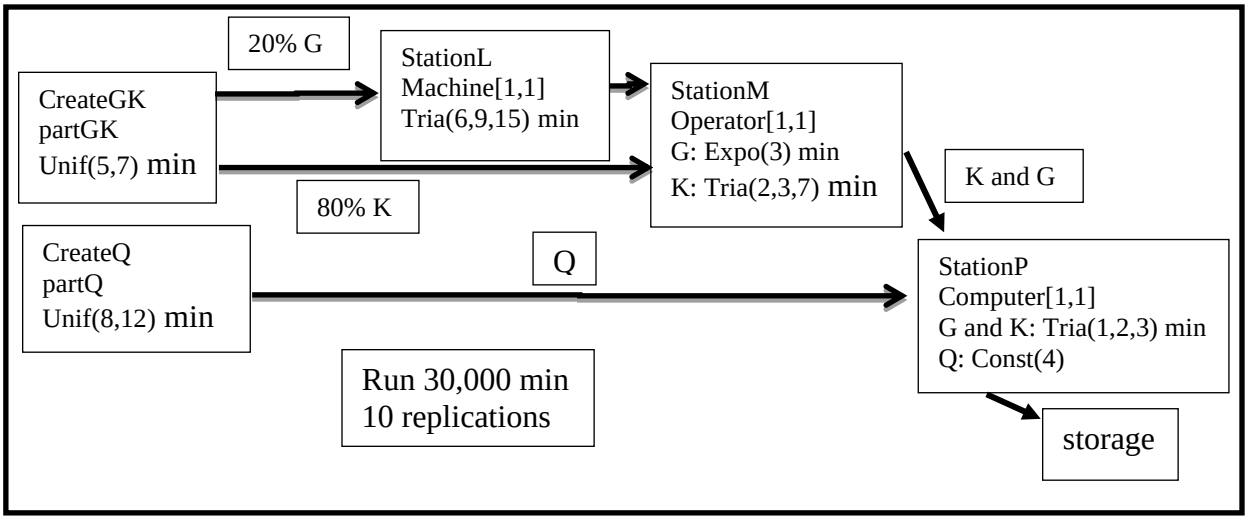


Figure 3: Logical Model of Example 2

This example showcases another valuable application of the Logical Model: facilitating utilization estimation through using a Logical Model with average values.

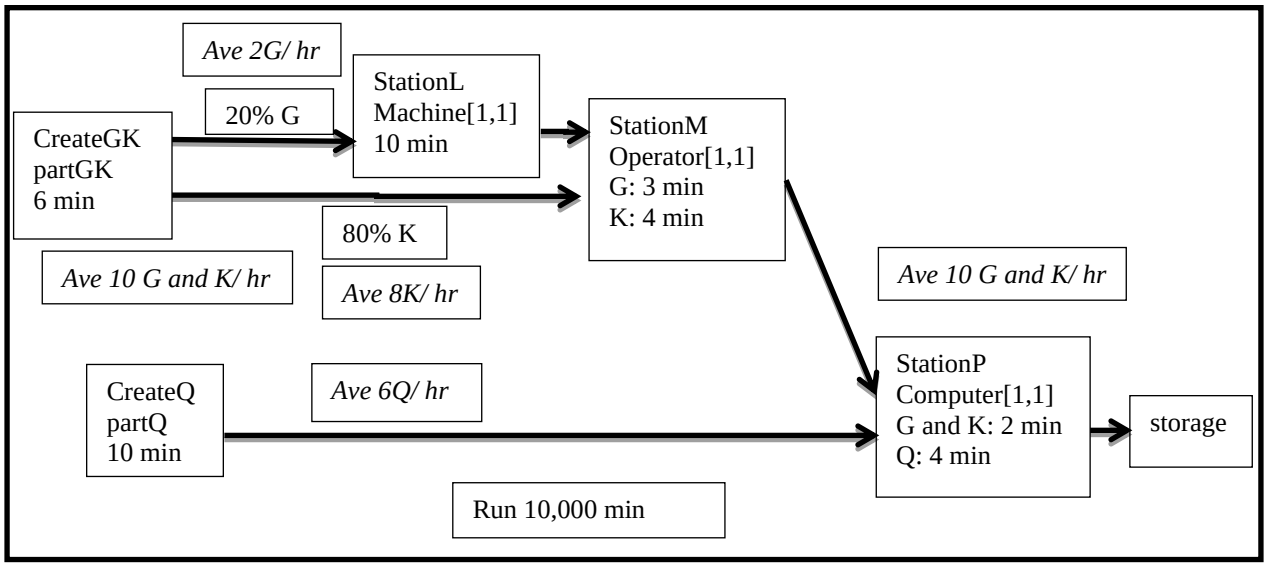


Figure 4: Logical Model of Example 2 with average values of distributions

Estimate utilization of Machine at Station L: Consider a subjective study period of one hour for this calculation. Parts GK arrive at the system at an average rate of 10 parts per hour (60 minutes / 6 minutes/part). Since 20% of parts are parts G, an average of 2 parts G enter Station L each hour. The service time per part G averages 10 minutes. Then, on average 6 parts G can be served during one hour is (60 minutes)/ (10 minutes)= 6 G. The. The utilization can be calculated as the average number of parts G arriving at Station L during one hour divided by the average number

of parts G that can be processed in one hour. Therefore, the estimated utilization is 0.333 (2 / 60).

Estimate utilization of Operator at Station M: An average of 2 parts G (each requiring 3 minutes of service) and 8 parts K (each requiring 4 minutes of service) arrive at Station M per hour. The required total service time during one hour is estimated as: (2 parts G * 3 minutes/part) + (8 parts K * 4 minutes/part) = 38 minutes/hour. The Operator (resource) is available for 60 minutes in one hour. The utilization can be calculated as average total service time needed in one hour divided by average available time of resources in one hour. Therefore, the estimated utilization is 0.633 (38 minutes/hour / 60 minutes/hour).

An alternative approach to estimate utilization of Operator leverages weighted average service time. Since the station only handles parts G and K, the average service time is a weighted average of their individual service times based on arrival probability: (0.2 * 3 minutes/part) + (0.8 * 4 minutes/part) = 3.8 minutes/part. The average inter-arrival time is 6 minutes. Utilization can be calculated as average weighted service time divided by average inter-arrival time. The estimated utilization remains 0.633 (3.8 minutes/part / 6 minutes/part). This demonstrates that both methods can lead to the same result.

Estimate utilization of Computer at Station P: Consider a subjective study period of one hour for this calculation. On average, 10 of parts K or G approach Station P in each hour. Each part K or G requires 2 minutes. On average, 6 of parts Q approaches Station P in each hour. Each part Q requires 4 minutes. The total average service time, required in one hour, is calculated as (10 parts * 2 minutes/part) + (6 parts * 4 minutes/part) = 44 minutes/hour.
Computer (resource) is available for 60 minutes in one hour. The utilization can be calculated as average total service time needed in one hour divided by average available time of resources in one hour. Therefore, the estimated utilization is 0.733 (44 minutes/hour / 60 minutes/hour).

The statistical output of this model is in Appendix C. The required values for verification are highlighted in the Appendix.

Table 2 summarizes the comparison between the estimated values and the values obtained from the statistical output of the Arena model. Considering the given threshold of 2% for the maximum acceptable percentage difference, the Arena model can be considered verified.

Table 2: Summary Table for Verification of Example 2

Data	Statistical output	Estimation	% difference
Utilization of Station L, Machine	0.332	0.333	0.3%
Utilization of Station M, Operator	0.634	0.63	0.6%
Utilization of Station P, Computer	0.734	0.733	0.14%

Notice that different approaches are presented here for estimating utilization. This assists to select the approach that is more suitable for communication at each situation.

6. Example 3: Verification of branching probability

This example focuses on verifying the branching probability implemented within the simulation model. For the sake of brevity, other verification aspects are not covered here.

Problem statement: Packing station with branching

Parts of two types, A and B, arrive at a Packing station individually. Their inter-arrival times are exponentially distributed with average intervals of 10 minutes (Expo(10)) for parts A and 12 minutes (Expo(12)) for parts B. Both part types queue together without priority for packing by a single Packer. The packing process itself takes an exponentially distributed time with an average of 3 minutes (Expo(3)) for all parts. Historically, 10% of all packed parts (A or B) are identified as defective and diverted to a Scrap storage area. The remaining 90% of packed parts are sent to the Warehouse.

a) Create a Logical model
b) Simulate this system for 50,000 unit of time and 10 replications
c) Verify the model using information on probability branching; consider a 2% threshold for verification.

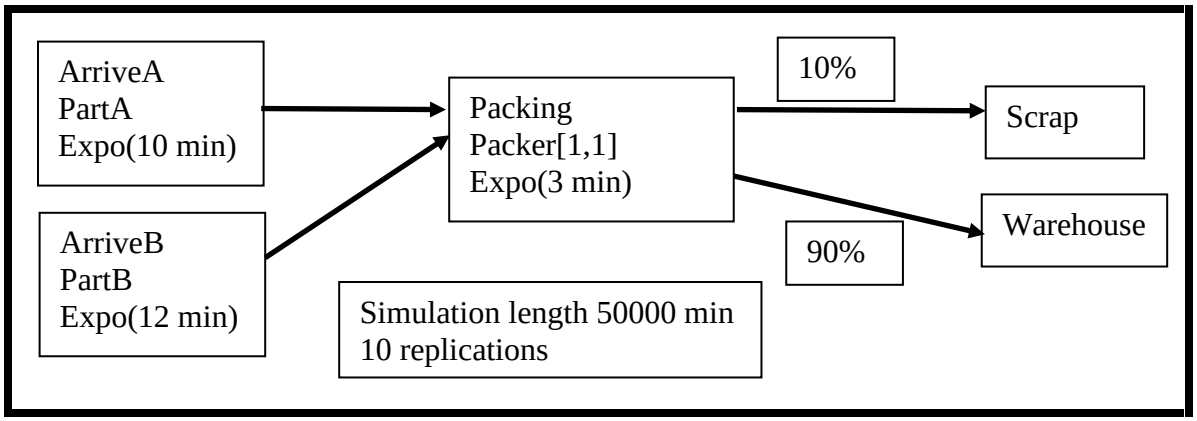


Figure 5: Logical Model of Example 3

To verify the proportion of packed parts reaching the warehouse, two temporary modules are added within the simulation model. In fact, only one of them is necessary. These modules are not intended for the final model but serve a verification purpose.

- A temporary module named "Record no Ware" (refer to Appendix D) is added to the path leading from Packing to the Warehouse. This module tracks the number of parts (both A and B) that successfully bypass the Scrap area and proceed to the Warehouse. This count is stored in a variable named "countWare" (refer to Appendix D for output).
- Similarly, a temporary module named "Record no Scrap" is added to the path leading from Packing to the Scrap. A temporary counter named "countScrap" captures the total number of parts diverted to Scrap.

Using the statistical output from the simulation in Appendix D, we obtained the following relevant data:

- Number of Part A leaving the system during simulation: 5,048
- Number of Part B leaving the system during simulation: 4,174.8
- Number of Parts (A & B) exiting the Warehouse during simulation: 8,300.6

Then, calculate the percentage of parts (A and B) reaching the warehouse:
(8,300.6 / (5,048 + 4,174.8)) = 0.9

It is important to note that we could have alternatively used the "countScrap" data point (number of parts entering Scrap) for this verification instead of relying on the warehouse exit count. This approach would provide the same conclusion.

The Arena software displays the count of entities leaving each branch at the end of the simulation run on the monitor. However, this method doesn't provide a well-documented record for future reference. Utilizing temporary modules within the model and capturing data statistically offers a more reliable and verifiable approach.

Table 3: Summary Table for Verification of Example 3

Data	Calculated based on Statistical output	Given info	% difference
Probability leaving for Warehouse	0.9	90%	0%

Assuming a difference of up to 2% is acceptable, the probability branching is verified.

7. Example 4: Verification of balk rate and critical thinking in verification process

This example focuses on verifying the balk rate (rate at which parts are turned away due to a full queue) within the simulation model. It also highlights the importance of critical thinking in applying other insights or topics, such as queuing theory. Unlike typical verification processes, this example does not utilize percentage differences or threshold values. Here the concept of upper bound or lower bound values are used for verification metrics. Such scenarios are less common in verification but still require careful consideration.

Problem statement: Packing station with limited queue capacity

Parts of type A arrive at a Packing station individually. Their inter-arrival times follow an exponential distribution with an average of 5 minutes (Expo(5)). Inside the Packing station, a single Operator performs a service with a normal distribution (Norm(3, 1) minutes) for processing time. However, the Packing station has a limited queue capacity that can only hold a maximum of 2 of parts A waiting for service. Any parts arriving when the queue is full will be overflowed (balked) and diverted to a temporary Storage area. Parts that successfully complete packing are then sent to the Warehouse.

a) Create a Logical model
b) Simulate this system for 30,000 minutes and 10 replications
c) Verify the model as much as possible

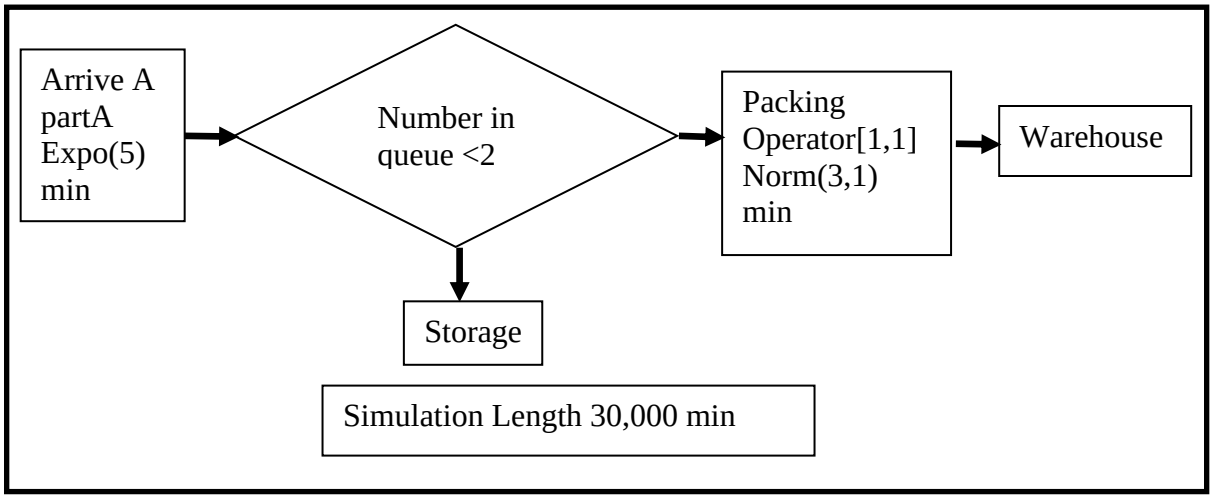


Figure 6: Logical Model of Example 4

Concepts from queuing theory can be applied to further verify the simulated model's behavior. If we replace the normal distribution (Norm(3, 1)) for service time with an exponential distribution (Expo(3)), the system can be modeled as an M/M/1/C queuing system. Refer to Figure 7 for the correspondence between the simulated model and queuing theory. Notably, the limited queue capacity of 2 in the simulation model aligns with the system capacity of C=3 in the M/M/1/C queuing model. Appendix F provides the relevant queuing theory formulas and detailed calculations in relation to required information for later verification.

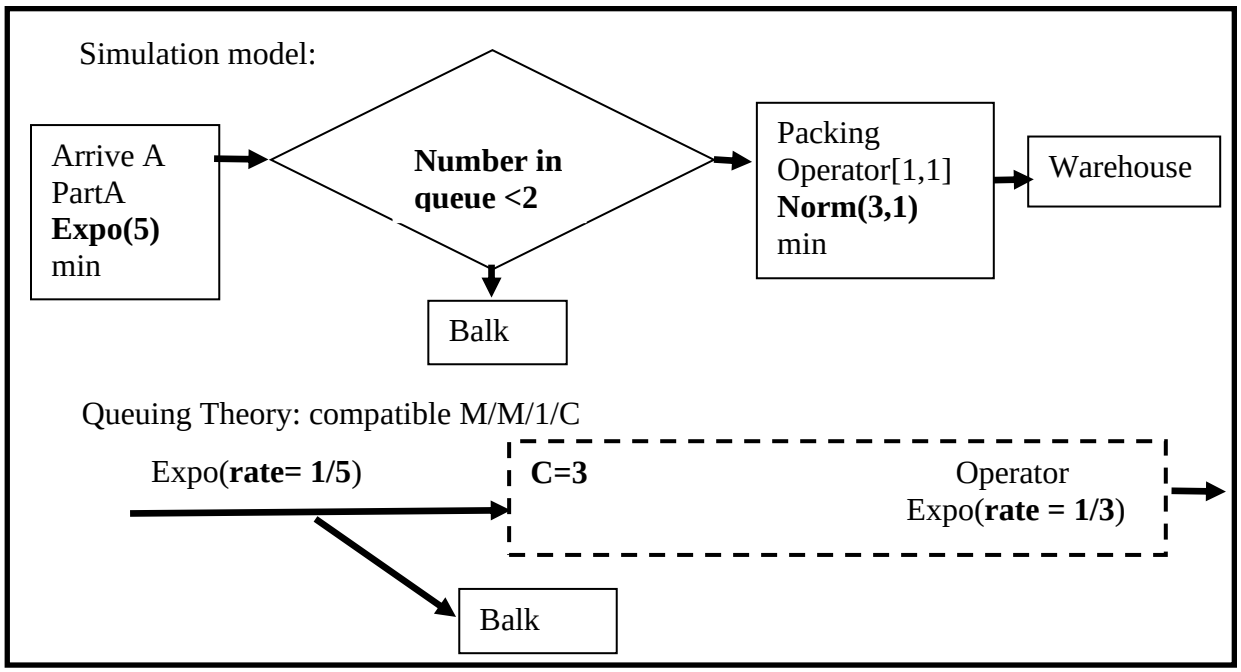


Figure 7: Queuing System Compatible with the Simulated Model

Impact of service time distribution on performance:
Since the average service times for the packing process in both the simulated model (Norm(3, 1)) and the queuing model (Expo(rate=1/3)) are identical, we can anticipate how the distribution difference will affect the system's behavior. The key difference lies in the variation, standard deviation. The exponential distribution (M/M/1/C queuing model) has a higher standard deviation (3) compared to the normal distribution in simulated model (1). This higher standard deviation in the exponential distribution has the following consequences:

- Increased queue length: Parts will experience more variable service times, leading to a higher average number of parts waiting in line (Lq).
- Increased waiting time: As a consequence of more parts queuing, the average waiting time in line (Wq) will also be higher.
- Potential for more balking: With a higher chance of encountering a full queue due to variable service times, we might expect an increase in the number of parts balking (turned away).
- Reduced Operator utilization: Since fewer parts will successfully enter the packing station for service due to balking, the operator's utilization is likely to be lower compared to the simulated model.

Calculations in relation to balking using simulation output in Appendix E:
Appendix E presents the statistical output from the simulation model, with relevant data highlighted. We can extract the following information (averaged over 10 replications):

- Total parts A entered: 6,031.8
- Total parts A balked: 320.4
- Simulation length: 30,000 minutes

The percentage of parts A balking can be calculated as:
Balk percentage = (Number of parts balked) / (Total parts entered) = (320.4) / (6,031.8) = 5.31%
This value represents the proportion of arriving parts A that were turned away due to a full queue. It is important to distinguish this from the balk rate itself, which is expressed as the rate (occurrences per unit time) at which balking happens. In this case, the balk rate can be calculated as:
Balk rate (per minute) = (Number of parts balked) / (Simulation length) = (320.4) / (30,000 minutes) = 0.011 parts per minute

Verification using Queuing Theory:
Table 4 presents the results from the simulation model (Appendix E) alongside the values predicted based on queuing theory (Appendix F). As indicated in the comment column, the performance measures (Lq, Wq, Operator Utilization) all align with our expectations. This agreement between the simulated behavior and the theoretical predictions from the M/M/1/C model provides good enough verification of the model, especially considering the non-identical service time distributions. It is important to note that this verification approach doesn't rely on the concept of percentage differences and thresholds typically used when comparing two identical models.

Table 4: Summary verification table for Example 4

	Simulated model	M/M/1/C	Comment
Source of data	Appendix E	Appendix F	
Service time distribution	Norm(3,1)	Expo(rate =1/3)	
Performance measures			
utilization	0.571	0.545	M/M/1/C is lower as expected
Average waiting time in line, min	1.49	2.06	M/M/1/C is higher as expected
Average number in line	0.28	0.37	M/M/1/C is higher as expected
Balk rate, number per min	0.011*	0.02	M/M/1/C is higher as expected

*Total number balked / simulation run length

8. Conclusion and future work

This paper introduces the Logical Model, a simple yet powerful tool to facilitate teaching several skills to students who take a discrete-event simulation course. The model offers a graphical representation of problem information, focusing on entity flow. Appendix A exemplifies its application for initial system analysis before model creation. Logical Model enables early problem identification and correction. It also leads to fewer semantic mistakes during model building.

This paper's primary focus is providing guidelines and examples for verifying simulation model output. Logical Model assists in estimating performance measures, which are then compared with the simulation's output values. The examples also showcase various approaches to resource utilization estimation.

The author received initial positive feedback on this approach around 2000. Encouraged by these observations, the author continued to utilize and refine the pedagogy. Here are suggestions for instructors interested in assessing the effectiveness of these tools:

1. Pre- and Post-Assessment Quiz: Provide a problem statement like example in Appendix A as a quiz before introducing the Logical Model or verification concepts. Limit the completion time. After covering these topics, repeat the quiz. Compare the time taken and correctness of students' models to gauge the impact of pedagogy.
2. Exam Comparison: After teaching the Logical Model, administer an exam question from a previous year. Compare the completion time and correctness of students' models with historical data to assess the pedagogy's effectiveness.

For further brainstorming on assessment methods, please contact the author at parisay@usc.edu.

Discrete-event simulation education encompasses more than just verification. Future papers will address additional areas requiring focus, such as:

- Application and appreciation of attributes
- Modeling and verifying various resource configurations
- Analysis and verification of breakdowns and failures

The Logical Model can be expanded to incorporate information about process variability. This advanced version proves valuable for output analysis and systematic improvement suggestions. Additionally, the author has developed tables used in conjunction with the Logical Model to facilitate systematic design of experiments. This approach guides students towards structured experimentation rather than random trial and error.

The Logical Model serves as a valuable tool for enhancing students' critical thinking and analytical skills. It empowers them to develop a deeper understanding of their models and move beyond simply trusting software outputs.

Appendix A: Example of a model blowing up

Purpose: This example highlights the importance of a thorough initial system analysis before diving into simulation modeling. Students are presented with a problem statement describing a subsystem for parts processing. The goal is not to build the simulation model itself, but to grapple with the challenges of modeling a flawed (unstable) system. Students work on the problem statement and encounter difficulties in creating a functional model. This struggle serves as a valuable learning experience, prompting a discussion on the critical role of initial analysis. Absence of an initial analysis is often a point of difficulty in student projects, mirroring challenges observed in many classroom models.

Problem statement: We are studying a subsystem for parts processing. Parts arrive individually, with an average interarrival time of six minutes following an exponential distribution. A single Machine at Station L processes each Part with a processing time drawn from a triangular distribution with a minimum, mode, and maximum of 10, 13, and 19 minutes respectively. Twenty percent of the Parts are then directed to Storage. The remaining Parts proceed to Station M, where a single Operator performs a process with a processing time for each Part uniformly distributed between 3 and 10 minutes. Finally, all Parts are sent to the same Storage location. A constant travel time of five minutes is assumed between stations, stations and Storage, and the entrance to Station L.

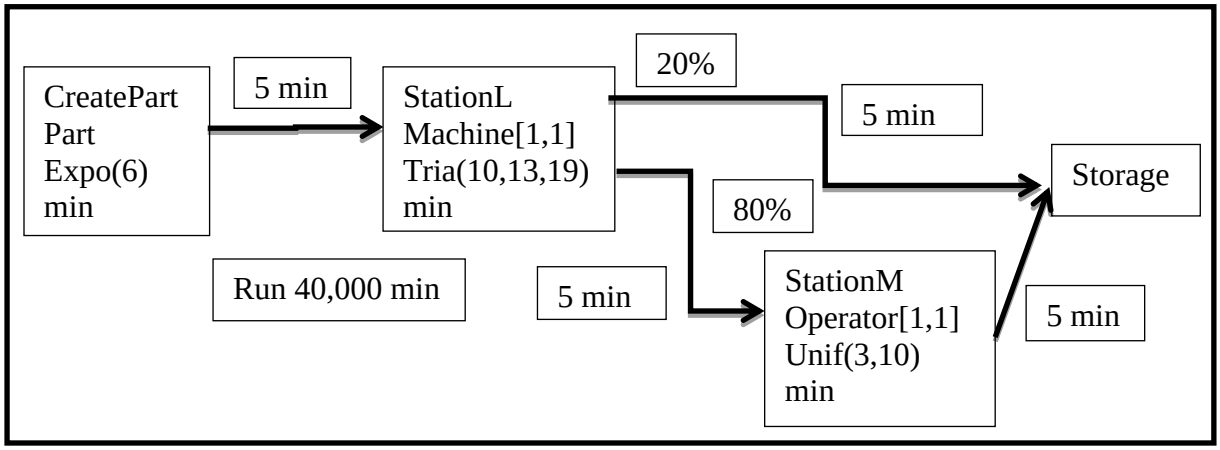


Figure A-1: Logical Model of the problem

Note: This problem replicates Example 1 with a crucial difference: the number of Machines at Station L is reduced from three to one. This change significantly impacts system stability.

Initial analysis to estimate utilization of the Machine at Station L: By analyzing the logical model, we can estimate utilization at Station L. Parts arrive on average every 6 minutes (exponential distribution average: 6) and require an average processing time of 14 minutes (triangular distribution average: (10+13+19)/3). Utilization can be calculated as average service time divided by average inter-arrival time. In this case, the estimated utilization for Station L is a concerning 2.33 (14/6), exceeding 1. This indicates an **unstable** system prone to queue buildup at Station L over time.

This initial analysis highlights the need for corrective action at Station L before diving into simulation software. Given the high utilization of 2.33, adding two additional Machines (for a total of three) is recommended to prevent system blowing up. Assume an acceptable utilization target is between 0.8 and 0.95. With three machines, each Machine's utilization would be 0.778 (2.33/3), which is considered under-utilized. Two Machines, however, would still result in an unstable system (2.33/2 = 1.16). Therefore, increasing the number of Machines to three offers a solution, even if it means accepting a lower individual Machine utilization.

Without this initial analysis, students often struggle to identify the root cause of issues during simulation. Running the model, whether the student or professional version, wouldn't directly reveal the need for three Machines. Students might observe queue buildup through animation, but without further exploration (potentially involving trial and error), they wouldn't quickly reach the optimal solution of three Machines. Trial and error in such cases is inefficient and time-consuming.

Running the model with the student version of Arena encounters a limiting factor: an error message appears when the entity limit of 150 is reached, and provides no output (Figure A-2). Notably, this message doesn't pinpoint the problem location or the underlying instability. Instructors advise running the model for a shorter duration (less than the stop time 1512.36) to obtain some output for analysis to pinpoint the issue. However, even this approach is less efficient compared to initial analysis of the logical model.

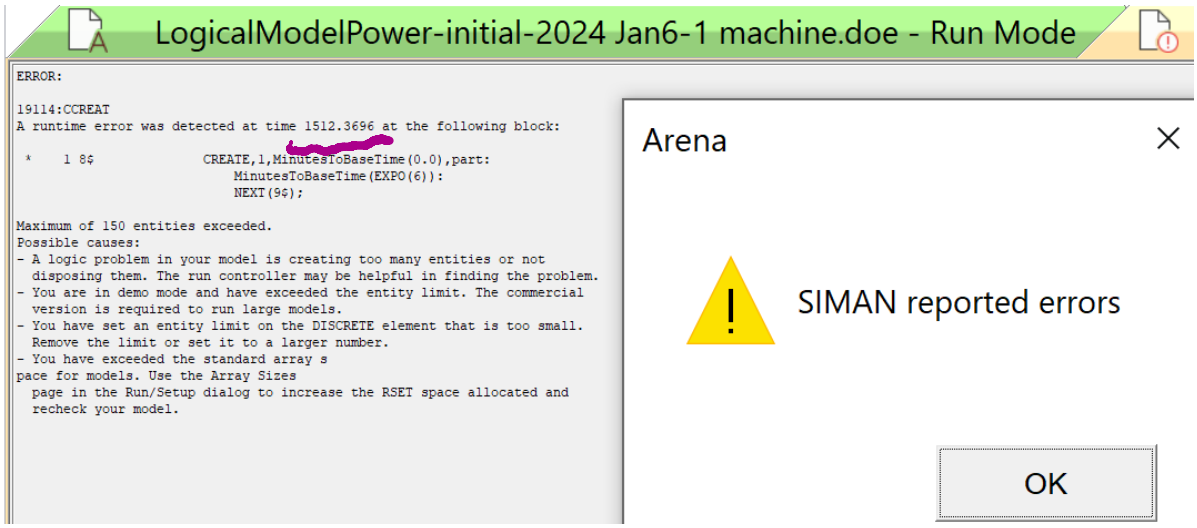
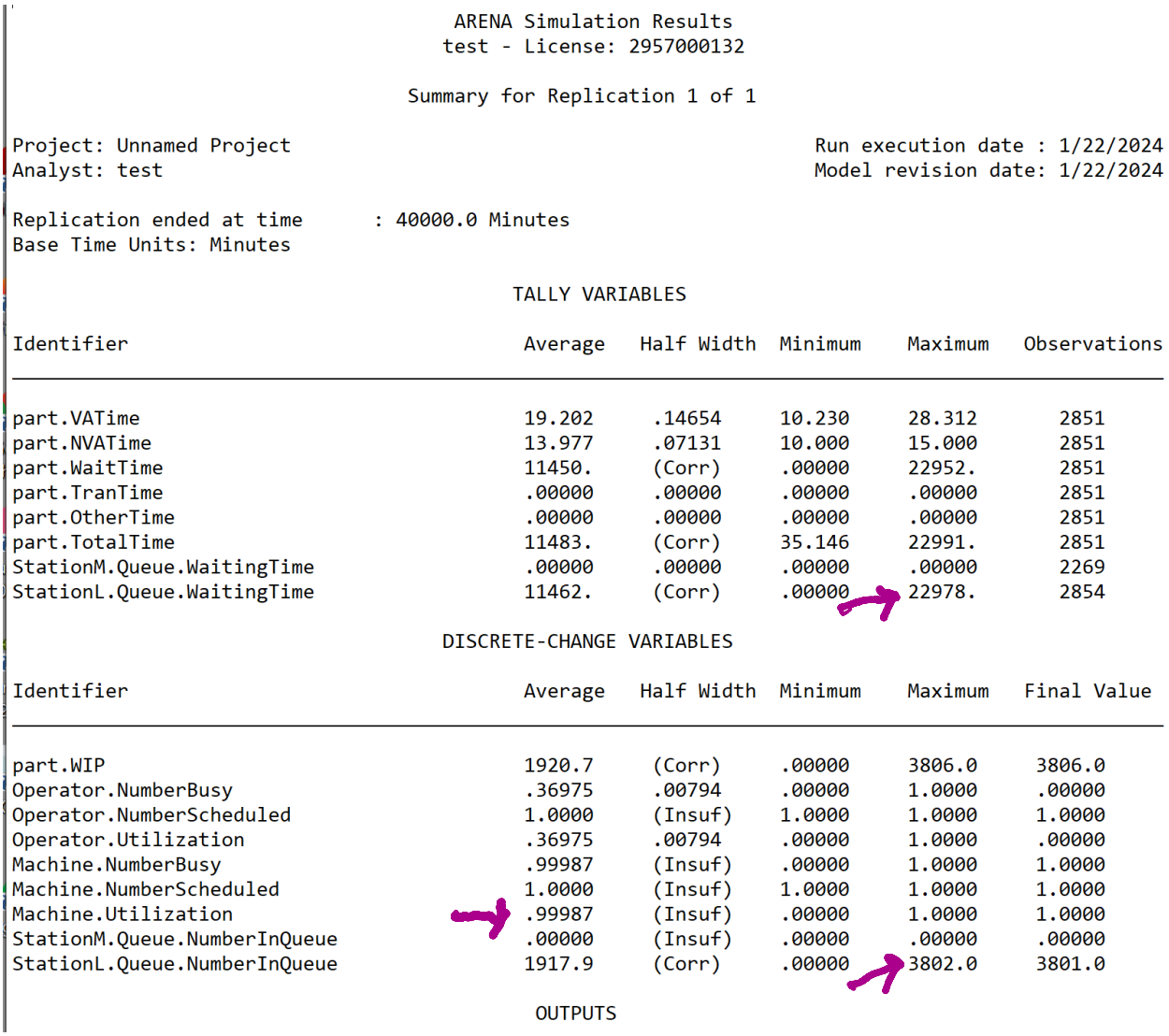


Figure A-2: Error message by the Student Version of Arena

Even with the professional version of Arena (Figure A-3 and A-4), most students struggled to identify the problem from the output alone. The data points that can lead to a blowup conclusion are marked in the following output. While the data does contain indicators of instability, such as very high machine utilization and unusually high values for maximum waiting time and number waiting in line, students often missed these subtle clues.

Note: The new version of Arena for the professional version I used lacked the "Category View" available for output in other appendices. Therefore, I've included the equivalent SIMAN output.



ARENA Simulation Results
test - License: 2957000132

Summary for Replication 1 of 1

Project: Unnamed Project — Run execution date : 1/22/2024
Analyst: test — Model revision date: 1/22/2024

Replication ended at time : 40000.0 Minutes
Base Time Units: Minutes

TALLY VARIABLES

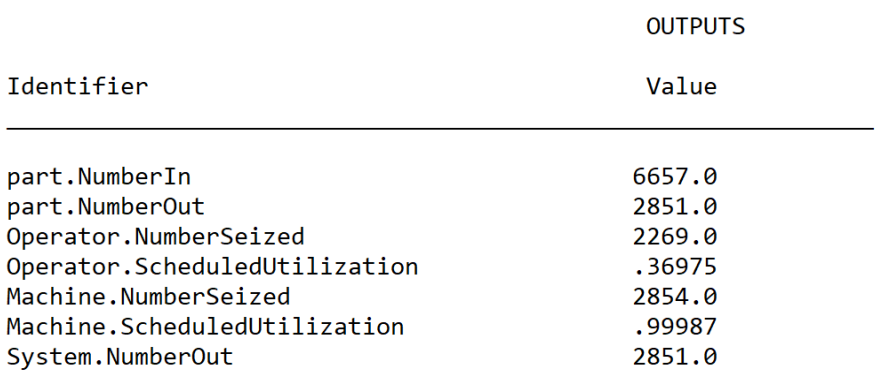
Identifier	Average	Half Width	Minimum	Maximum	Observations
part.VATime	19.202	.14654	10.230	28.312	2851
part.NVATime	13.977	.07131	10.000	15.000	2851
part.WaitTime	11450.	(Corr)	.00000	22952.	2851
part.TranTime	.00000	.00000	.00000	.00000	2851
part.OtherTime	.00000	.00000	.00000	.00000	2851
part.TotalTime	11483.	(Corr)	35.146	22991.	2851
StationM.Queue.WaitingTime	.00000	.00000	.00000	.00000	2269
StationL.Queue.WaitingTime	11462.	(Corr)	.00000	22978.	2854

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
part.WIP	1920.7	(Corr)	.00000	3806.0	3806.0
Operator.NumberBusy	.36975	.00794	.00000	1.0000	.00000
Operator.NumberScheduled	1.0000	(Insuf)	1.0000	1.0000	1.0000
Operator.Utilization	.36975	.00794	.00000	1.0000	.00000
Machine.NumberBusy	.99987	(Insuf)	.00000	1.0000	1.0000
Machine.NumberScheduled	1.0000	(Insuf)	1.0000	1.0000	1.0000
Machine.Utilization	.99987	(Insuf)	.00000	1.0000	1.0000
StationM.Queue.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
StationL.Queue.NumberInQueue	1917.9	(Corr)	.00000	3802.0	3801.0

OUTPUTS

Figure A-3: Output by the Professional Version of Arena

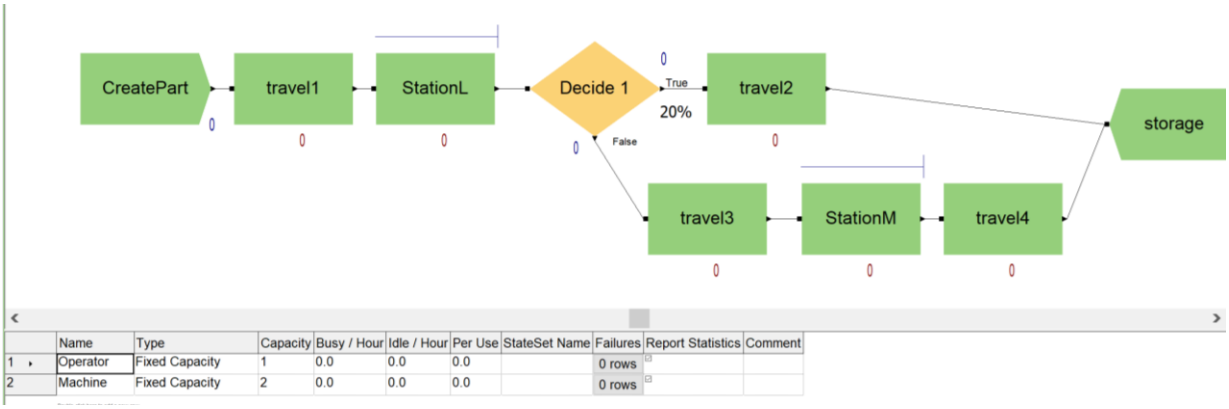


OUTPUTS

Identifier	Value
part.NumberIn	6657.0
part.NumberOut	2851.0
Operator.NumberSeized	2269.0
Operator.ScheduledUtilization	.36975
Machine.NumberSeized	2854.0
Machine.ScheduledUtilization	.99987
System.NumberOut	2851.0

Figure A-4: Output by the Professional Version of Arena, continued

Appendix B: Example 1 Statistical Output



	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics	Comment
1	Operator	Fixed Capacity	1	0.0	0.0	0.0		0 rows		
2	Machine	Fixed Capacity	2	0.0	0.0	0.0		0 rows		

Figure B-1: Screenshot of Arena

Arena's output, averages of 10 replications:

8:05:07PM **Category Overview** January 5, 2024

Values Across All Replications

InitialLogicalModel

Replications: 10 Time Units: Minutes

Entity

Time

VA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	19.2038	0.04	19.1172	19.2981	10.0232	28.7339

NVA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	14.0020	0.02	13.9556	14.0359	10.0000	15.0000

Wait Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	19.7039	2.38	16.2147	26.0344	0.00	196.09

Transfer Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	0.00	0.00	0.00	0.00	0.00	0.00

Other Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	0.00	0.00	0.00	0.00	0.00	0.00

Total Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	52.9097	2.42	49.3419	59.3461	20.0808	232.65

Other

Number In	Average	Half Width	Minimum Average	Maximum Average
part	6608.50	56.88	6486.00	6744.00

Number Out	Average	Half Width	Minimum Average	Maximum Average
part	6599.40	54.40	6479.00	6721.00

WIP	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
part	8.7413	0.45	7.9993	9.7847	0.00	36.0000

Figure B-2: Output by the Student Version of Arena

8:05:07PM **Category Overview** January 5, 2024

Values Across All Replications

InitialLogicalModel

Replications: 10 Time Units: Minutes

Queue

Time

Waiting Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
StationL.Queue	6.2376	0.50	5.2555	7.3067	0.00	66.4251
StationM.Queue	16.8166	2.39	13.4021	23.2030	0.00	173.02

Other

Number Waiting	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
StationL.Queue	1.0311	0.09	0.8521	1.2051	0.00	15.0000
StationM.Queue	2.2275	0.33	1.7681	3.0875	0.00	26.0000

8:05:07PM **Category Overview** January 5, 2024

Values Across All Replications

InitialLogicalModel

Replications: 10 Time Units: Minutes

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Machine	0.7704	0.01	0.7519	0.7875	0.00	1.0000
Operator	0.8597	0.01	0.8432	0.8865	0.00	1.0000

Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Machine	2.3111	0.02	2.2557	2.3626	0.00	3.0000
Operator	0.8597	0.01	0.8432	0.8865	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Machine	3.0000	0.00	3.0000	3.0000	3.0000	3.0000
Operator	1.0000	0.00	1.0000	1.0000	1.0000	1.0000

Scheduled Utilization	Average	Half Width	Minimum Average	Maximum Average
Machine	0.7704	0.01	0.7519	0.7875
Operator	0.8597	0.01	0.8432	0.8865

8:05:07PM **Category Overview** January 5, 2024

Values Across All Replications

InitialLogicalModel

Replications: 10 Time Units: Minutes

Resource

Usage

Total Number Seized	Average	Half Width	Minimum Average	Maximum Average
Machine	6606.80	56.15	6485.00	6740.00
Operator	5283.70	54.96	5189.00	5427.00

Figure B-3: Output by the Student Version of Arena

Appendix C: Example 2 Statistical Output

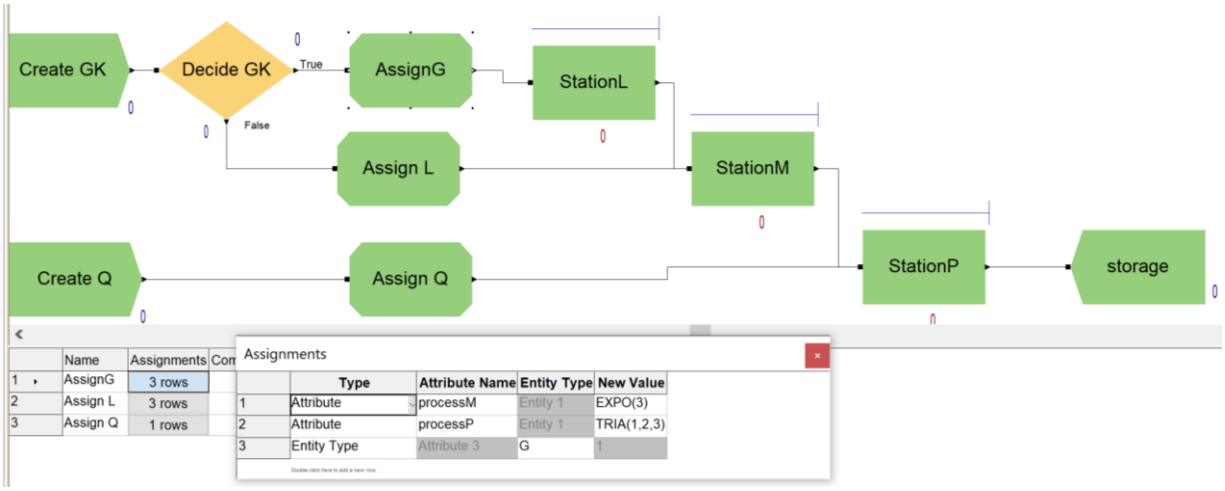


Figure C-1: Screenshot of Arena

Arena's partial output, averages of 10 replications:

10:17:17PM **Category Overview** January 5, 2024

Values Across All Replications

VM2verification

Replications: 10 Time Units: Minutes

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
computer	0.7340	0.00	0.7321	0.7357	0.00	1.0000
machine	0.3323	0.00	0.3252	0.3383	0.00	1.0000
operator	0.6342	0.00	0.6303	0.6453	0.00	1.0000

Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
computer	0.7340	0.00	0.7321	0.7357	0.00	1.0000
machine	0.3323	0.00	0.3252	0.3383	0.00	1.0000
operator	0.6342	0.00	0.6303	0.6453	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
computer	1.0000	0.00	1.0000	1.0000	1.0000	1.0000
machine	1.0000	0.00	1.0000	1.0000	1.0000	1.0000
operator	1.0000	0.00	1.0000	1.0000	1.0000	1.0000

Scheduled Utilization	Average	Half Width	Minimum Average	Maximum Average
computer	0.7340	0.00	0.7321	0.7357
machine	0.3323	0.00	0.3252	0.3383
operator	0.6342	0.00	0.6303	0.6453

Figure C-2: Output by the Student Version of Arena

Appendix D: Example 3 Statistical Output

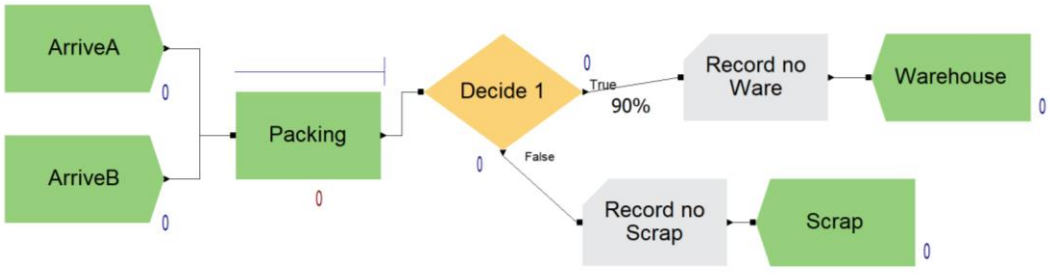
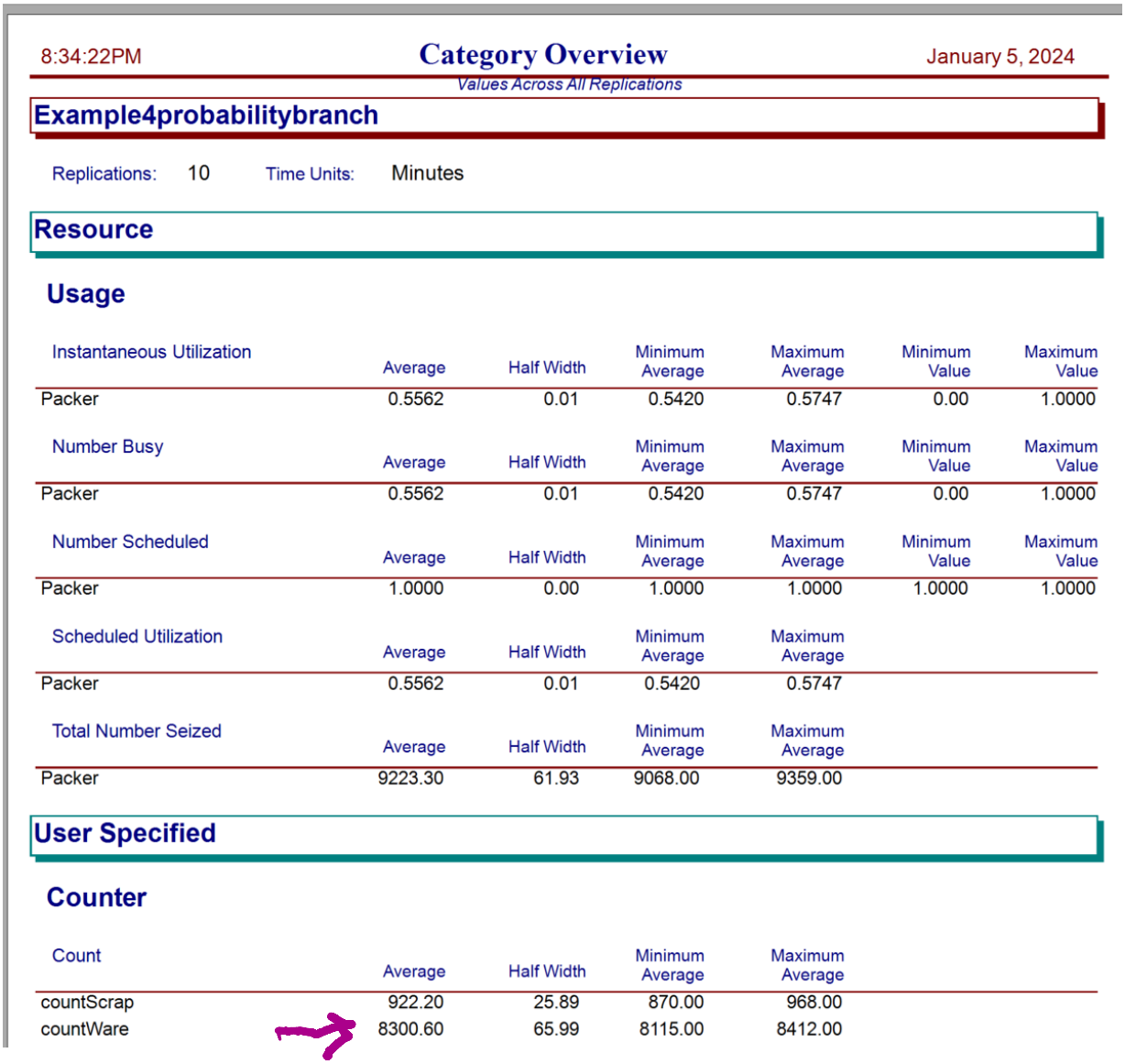


Figure D-1: Screenshot of Arena

Arena's partial output, averages of 10 replications:



8:34:22PM **Category Overview** January 5, 2024

Values Across All Replications

Example4probabilitybranch

Replications: 10 Time Units: Minutes

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packer	0.5562	0.01	0.5420	0.5747	0.00	1.0000

Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packer	0.5562	0.01	0.5420	0.5747	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packer	1.0000	0.00	1.0000	1.0000	1.0000	1.0000

Scheduled Utilization	Average	Half Width	Minimum Average	Maximum Average
Packer	0.5562	0.01	0.5420	0.5747

Total Number Seized	Average	Half Width	Minimum Average	Maximum Average
Packer	9223.30	61.93	9068.00	9359.00

User Specified

Counter

Count	Average	Half Width	Minimum Average	Maximum Average
countScrap	922.20	25.89	870.00	968.00
countWare	8300.60	65.99	8115.00	8412.00

Figure D-2: Output by the Student Version of Arena

8:34:22PM **Category Overview** January 5, 2024

Values Across All Replications

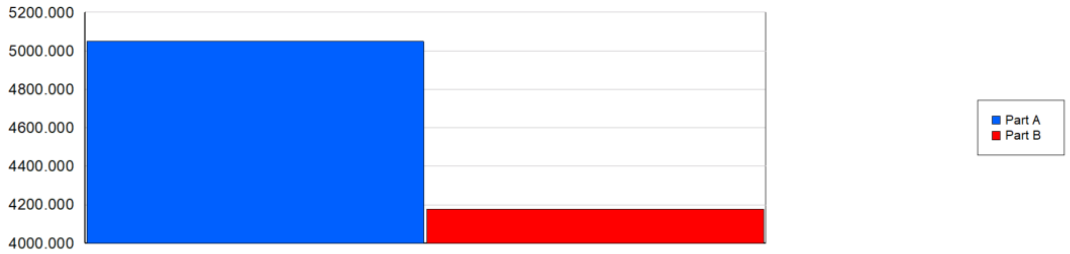
Example4probabilitybranch

Replications: 10 Time Units: Minutes

Entity

Other

Number In	Average	Half Width	Minimum Average	Maximum Average
Part A	5048.90	55.88	4911.00	5174.00
Part B	4175.40	38.79	4098.00	4285.00



Number Out	Average	Half Width	Minimum Average	Maximum Average
Part A	5048.00	55.91	4910.00	5174.00
Part B	4174.80	38.87	4098.00	4285.00

WIP	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Part A	0.6880	0.02	0.6445	0.7164	0.00	12.0000
Part B	0.5697	0.01	0.5330	0.5903	0.00	13.0000

Queue

Time

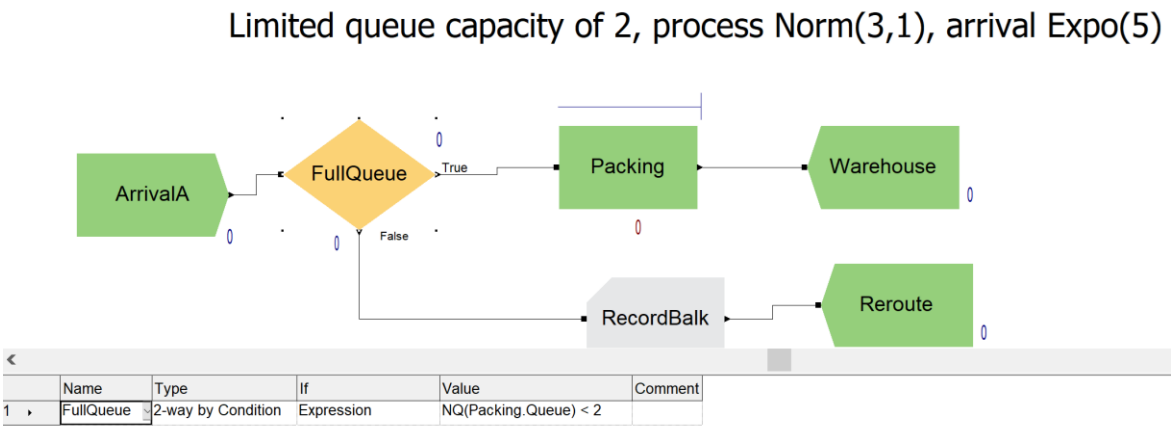
Waiting Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packing.Queue	3.8016	0.11	3.4830	3.9527	0.00	51.7021

Other

Number Waiting	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packing.Queue	0.7015	0.02	0.6418	0.7319	0.00	16.0000

Figure D-3: Output by the Student Version of Arena

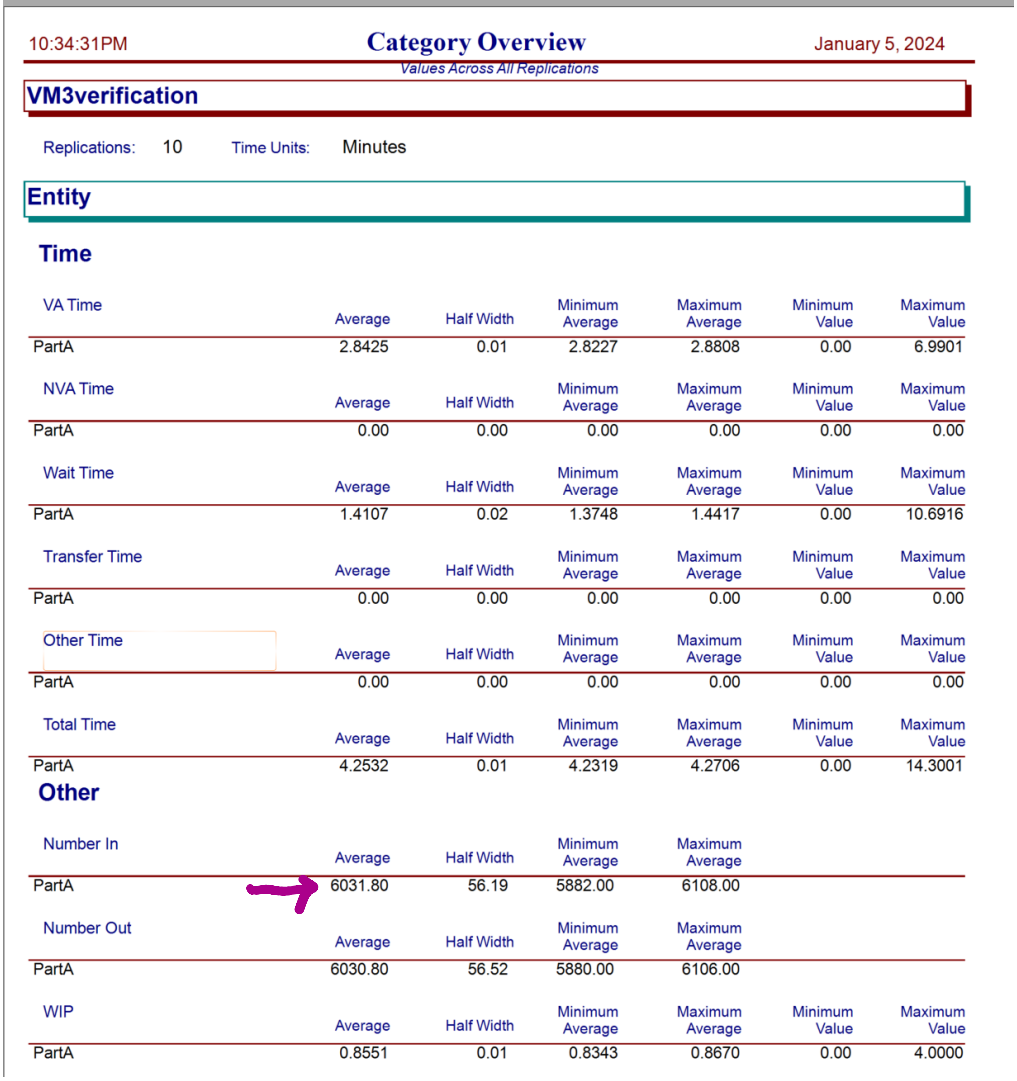
Appendix E: Example 4 Statistical Output



	Name	Type	If	Value	Comment
1	FullQueue	2-way by Condition	Expression	NQ(Packing.Queue) < 2	

Figure E-1: Screenshot of Arena

Arena's partial output, averages of 10 replications:



10:34:31PM **Category Overview** January 5, 2024

Values Across All Replications

VM3verification

Replications: 10 Time Units: Minutes

Entity

Time

VA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	2.8425	0.01	2.8227	2.8808	0.00	6.9901

NVA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	0.00	0.00	0.00	0.00	0.00	0.00

Wait Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	1.4107	0.02	1.3748	1.4417	0.00	10.6916

Transfer Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	0.00	0.00	0.00	0.00	0.00	0.00

Other Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	0.00	0.00	0.00	0.00	0.00	0.00

Total Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	4.2532	0.01	4.2319	4.2706	0.00	14.3001

Other

Number In	Average	Half Width	Minimum Average	Maximum Average
PartA	6031.80	56.19	5882.00	6108.00

Number Out	Average	Half Width	Minimum Average	Maximum Average
PartA	6030.80	56.52	5880.00	6106.00

WIP	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
PartA	0.8551	0.01	0.8343	0.8670	0.00	4.0000

Figure E-2: Output by the Student Version of Arena

10:34:31PM **Category Overview** January 5, 2024

Values Across All Replications

VM3verification

Replications: 10 Time Units: Minutes

Queue

Time

Waiting Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packing.Queue	1.4899	0.02	1.4371	1.5301	0.00	10.6916

Other

Number Waiting	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Packing.Queue	0.2837	0.01	0.2696	0.2931	0.00	2.0000

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
operator	0.5714	0.00	0.5610	0.5823	0.00	1.0000

Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
operator	0.5714	0.00	0.5610	0.5823	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
operator	1.0000	0.00	1.0000	1.0000	1.0000	1.0000

Scheduled Utilization	Average	Half Width	Minimum Average	Maximum Average
operator	0.5714	0.00	0.5610	0.5823

Total Number Seized	Average	Half Width	Minimum Average	Maximum Average
operator	5711.00	42.19	5628.00	5807.00

User Specified

Counter

Count	Average	Half Width	Minimum Average	Maximum Average
NumberBulk	320.40	22.94	253.00	362.00

Figure E-3: Output by the Student Version of Arena

Appendix F: Example 4, Using Queuing Theory

arrival rate per minute = $\lambda = 1/5 = 0.2$

service rate per minute = $\mu = 1/3 = 0.33$

Queue capacity = 2, then system capacity = C = 3

Using mathematical formula related to M/M/1/C system we will have the followings:

$\rho = 0.6$

$Probability\ of\ system\ empty = \pi_0 = (1-\rho)/(1-\rho^{C+1}) = 0.4554$

$Average\ number\ in\ server = L_s = 1 - \pi_0 = 0.5405$

$$Average\ number\ in\ system = L = \frac{\rho[1-(c+1)\rho^c + c\rho^{c+1}}{(1-\rho^{c+1})(1-\rho)}$$

$$= 0.9146$$

$Average\ number\ in\ line = L_q = L - L_s = 0.37$

$Probability\ of\ balking = \pi_C = \rho^C \pi_0 = 0.0993$

$$Average\ waiting\ time\ in\ line = W_q = \frac{L_q}{\lambda(1-\pi_C)} = 2.06$$

$Utilization = 1 - \pi_0 = 0.545$

$Balk\ rate = \lambda\pi_C = 0.02$

References

[1] R.G. Sargent and O. Balci, “History of Verification and Validation of Simulation Models,” in Proceedings of the 2017 Winter Simulation Conference, Las Vegas, Nevada, USA, December 3-6, 2017. pp. 292-307.

[2] W.D. Kelton, N. Zupick, and N. Ivey, *Simulation with Arena.* McGraw-Hill, 7th Edition, 2024.

[3] R.B. Whitner and O. Balci, “Guidelines for Selecting and Using Simulation Model Verification Techniques,” in Proceedings of the 1989 Winter Simulation Conference, Piscataway, New Jersey, 1989, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. 1989. pp. 559-568.

[4] O. Balci, “Verification, Validation and Accreditation of Simulation Models,” in Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, USA, December 7-10, 1997, S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, Eds. 1997, pp. 135–41.

[5] B. Ghosh, R. Bowden, B. Gladwin, and C. Harrell, *Simulation using ProModel.* Cognella Academic Publishing, 4th Ed, 2022.

[6] A. M. Law, *Simulation Modeling & Analysis*. McGraw-Hill, 6th Edition, 2024.

[7] R.G. Sargent, “Verification and Validation of Simulation Models: An Advanced Tutorial,” in Proceedings of the 2020 Winter Simulation Conference, Orlando, FL, USA, December 14-18, 2020. pp. 16-29.

[8] A.M. Law, “How to Build Valid and Credible Simulation Models,” in Proceedings of the 2022 Winter Simulation Conference (WSC), Singapore, December 11-14, 2022. pp. 1283-1295.

[9] B. Khoshnevis and S. Parisay, "Machine Learning and Simulation - Application in Queuing Systems," *Simulation*, vol. 61, no. 5, pp. 294-302, November 1993.

Acknowledgement

I would like to express my sincere gratitude to Ms. Neda Afsarmanesh for her invaluable feedback and insightful edits during the proofreading process. I am also deeply grateful to my students, whose insightful questions and challenges have been a constant source of inspiration, pushing me to refine my teaching methods to match my students’ needs.