# AN ABSTRACT OF THE THESIS OF

Benjamin Fields for the degree of Master of Science in Industrial Engineering presented on November 20, 2018.

Title: A Text-based Simulation Framework for the Automated Simulation and Analysis of Manufacturing Assembly Systems

Abstract approved: _____
J. David Porter

The rapid advancement of manufacturing has led to the creation of a myriad of technologies that facilitate the analysis and simulation of manufacturing processes. These technologies have become a pivotal component in maintaining and improving processes in today's complex manufacturing environments. Discrete Event Simulation (DES) is one such technology that has seen widespread use in the analysis of manufacturing assembly systems. However, despite its widespread use, organizations recognize that most of the commercially available DES software packages used to develop and maintain simulation models are costly and require significant levels of expertise, time, and resources.

This research introduces a new text-based simulation framework titled the Automated Simulation Analysis Engine (ASAE) that aims to reduce the steep learning curve typically experienced by users when creating, running, and analyzing a simulation model. Two testing approaches were used to validate the correctness and usability of the proposed test-based simulation framework. The results of these tests suggest that the ASAE simulation framework has the potential of reducing the skills, time, and resources required when conducting a simulation-based study.

# A Text-based Simulation Framework for the Automated Simulation and Analysis of Manufacturing Assembly Systems

by
Benjamin Fields

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented November 20, 2018
Commencement June 2019

Master of Science thesis of Benjamin Fields presented on November 20, 2018.

APPROVED:

Major Professor, representing Industrial Engineering

Minor Professor, representing Computer Science

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Benjamin Fields, Author

# ACKNOWLEDGEMENTS

While it may have seemed impossible at times, I am elated to be able to write today reflecting on this experience and the incredible journey I have taken along the way. It is no short feat that certainty would not have been possible without the endless love and support from my family, loved ones, close friends, and professors here at Oregon State. I would first like to express my appreciation and thanks for all the support I have received from my advisor Dr. J. David Porter. Without his guidance I would not be where I am today. I would also like to thank my family who have been there by my side every step of the way. I cannot express in words the gratitude that I have for having a dad to show me the door into the world of engineering, a mom to bring me back to reality when nothing seems to make sense, and the brothers to learn from each and every day. I am extremely grateful for all the professors that have helped me along the way and the friends I have made here at Oregon State. I will forever cherish my time here and look forward to the next chapter of my life as an OSU alum.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

# 1 INTRODUCTION

Today's manufacturing environments are constantly evolving as companies and researchers find new and innovative ways to create products. These constant changes have, in turn, created very competitive global markets which require highly complex manufacturing engineering and production management decisions. Technology has become a pivotal component in maintaining and improving processes in today's complex manufacturing environments. Programmatic techniques that incorporate computer programming technologies alongside integrated information systems are being used in manufacturing to accomplish tasks such as data processing and process simulation (Heavey & Robin, 2014).

In particular, discrete event simulation (DES) is one technology-based approach that has seen widespread use in industry to improve manufacturing processes. DES is defined as a broad collection of methods and applications to mimic the behavior of real systems with events occurring at distinct points in time (Kelton, Sadowski, and Zupick, 2010). However, despite its widespread use, organizations recognize that most of the commercially available DES software packages used to develop and maintain simulation models are costly and require significant levels of expertise, time, and resources (Hughes, Scott, & Ridgway, 2013).

The complexity of DES software packages has motivated interest in automated data-driven approaches that can reduce the amount of time and resources

needed to create and conduct simulations. Automated data-driven approaches streamline the analysis of a system by leveraging the information inherently contained within data. Process mining is one such data-driven technology that aims to discover, monitor, and improve processes by extracting knowledge from data contained in information systems (Aguirre, Parra, & Alvarado, 2013).

Researchers are beginning to recognize the benefits of merging simulation technologies and automated data-driven approaches to gain better and faster insight into processes. By combining DES approaches and data-driven technologies, the cost and complexity of simulation-based projects can potentially be reduced. The combination of these technologies can also be applied to complex manufacturing assembly systems that often involve a mix of dynamic resources and processes.

## 1.1 RESEARCH MOTIVATION

The modeling and analysis of a manufacturing assembly system (MAS) enabled by a DES software package is a valuable exercise to gain a better understanding of current process behavior and to identify opportunities for improvement. This is especially critical in the early stages of a simulation project when trying to develop accurate representations of a MAS. Unfortunately, the acquisition and licensing costs of DES software packages, the level of expertise they require, and the time and resources needed to maintain MAS models are significant barriers for their widespread use, particularly for small and medium size enterprises (Byrne, Liston, Geraghty, & Young, 2012).

Therefore, there is a growing need for low-cost, efficient, and adaptive DES technologies (Mourtzis, Doukas, & Bernidaki, 2014). In particular, recent research has focused on improving the current use of simulation technologies by automating the simulation and analysis steps, which in turn saves valuable time and resources that can be further utilized in more value-added activities (Bergmann & Strassburger, 2010).

## 1.2 Research Objective

The main objective of this research was to develop an automated, text-based simulation framework for modeling, running, and analyzing MASs. The proposed automated, text-based simulation framework aims to minimize the steep learning curve typically experienced by users when initially creating, running, and analyzing a simulation model with most of today's commercially available simulation software packages.

## 1.3 Research Contributions

The main contribution of this research was an automated, text-based simulation framework referred to as the Automated Simulation Analysis Engine (ASAE). The proposed ASAE text-based simulation framework includes the following features:

- A text-based modeling approach,
- Automated data collection, and
- Automated simulation and analysis of a MAS.

# 2 LITERATURE REVIEW

The main objective of this research was to develop a data-driven methodology to expedite the modeling, simulation, and analysis of manufacturing assembly systems by automating a large portion of the tasks normally required. With this objective in mind, a thorough literature review was conducted on several areas including process improvement in manufacturing, discrete event simulation software in manufacturing, and the intersection of data-driven analytical technologies with simulation. The relevant findings of the literature review are synthesized in this chapter.

The remainder of this chapter is organized as follows. Section 2.1 covers process improvement in manufacturing. Section 2.2 discusses the challenges with discrete event simulation software in manufacturing. Section 2.3 shows how data-driven technologies can be used to facilitate the analysis of simulation models. Finally, Section 2.4 summarizes the main findings of the literature review and clearly identifies the research gaps that this research attempts to fill.

## 2.1 PROCESS IMPROVEMENT IN MANUFACTURING

Manufacturing today involves the cohesion of multiple workstations into a systematic workflow to produce jobs for an increasingly growing global market. This has, in turn, created a highly competitive market with complex manufacturing engineering and production management decisions (Heavey & Robin, 2014). In

response to these challenges, researchers are beginning to adopt analytics-based approaches to keep up with continuous improvement, which have shown that favorable outcomes can be achieved with little investment.

One interesting trend is the combination of computer programming and data mining techniques. Recent research indicates that these integrative approaches to drive process improvement can help in improving the performance of manufacturing assembly systems and in reducing costs. Ham and Park (2014) proposed a framework to effectively balance manned assembly lines with an integrated video module written in the C++ programming language. Video recorded on the manned assembly line was reviewed to quickly and easily extract and transfer useful process information. The video review step was followed by the analysis of motion in the workstation, operation cycles, and workers' involvement within the workstation. The proposed framework was implemented and validated in a Korean assembly line for LED televisions. Similarly, Zheng et al. (2014) developed an integrated analytics system to improve the analysis and performance of a plasma display panel (PDP) manufacturing system. The integrated analytics system, referred to as PDP-Miner, is an example of a data-driven tool aimed at alleviating the burden of having engineering personnel spend large amounts of time and resources working through large data sets to create solutions to production problems. A case study at ChangHong COC, a large PDP manufacturer in China, showed that the implementation of PDP-Miner increased production levels by 10,000 units per month (i.e., a 3% increase in yield) which demonstrates that data-

driven process improvement techniques can save time while also improving the production system.

Researchers have also noted the need to develop specific methodologies that facilitate the implementation of strategies that incorporate technology into the redesign process. Intelligent redesign tools that build on current models, leverage a deeper understanding of the underlying process, and apply reproducible, objective transformations that can translate into time and labor savings. Aguirre, Parra, and Alvarado (2013) proposed a framework to successfully integrate process mining and simulation tools to drive process improvement. In the proposed framework, process mining complements simulation tools by extracting the parameters of an underlying model, which is then used to test alternative process designs. This approach to simulation provides a cost-effective method to test and validate new processes. The authors noted that the involvement of existing workforce is critical to correctly interpret activities and the data collected by the process mining systems, and that there is a need for better visualizations of simulation models.

## 2.2 DISCRETE EVENT SIMULATION SOFTWARE IN MANUFACTURING

Discrete event simulation (DES) is defined as a broad collection of methods and applications to mimic the behavior of real systems with events occurring at distinct points in time (Kelton, Sadowski, and Zupick, 2010). DES has been widely used in manufacturing for many years but, despite its widespread use, organizations recognize that most of the commercially available DES software packages used to

develop and maintain simulation models are costly and require significant levels of expertise, time, and resources (Hughes, Scott, & Ridgway, 2013). Although the licensing cost of DES software packages is a significant barrier, their complexity (i.e., the required steep learning curve) is often a challenge when incorporating the use of these modelling tools in industry projects, particularly in small to medium size enterprises (SMEs). Therefore, only a select group of individuals is really in a position to use DES software packages, thus minimizing the potential benefits that could be realized across the organization (Byrne, Liston, Geraghty, & Young, 2012).

A number of open source projects have appeared recently whose objective is to make DES software more accessible and to provide an easier method for creating simulation models. Rossetti (2008) describes an open source, object-oriented framework for creating simulation models within the Java programming language named the Java Simulation Library (JSL). The JSL contains four main packages (i.e., Utilities, Calendar, Modeling, and Observers) that facilitate the execution and creation of simulation models using an object-oriented approach that utilizes pre-built class objects that implement the functionality of simulation events and models. Byrne, Liston, Geraghty, & Young (2012) presented two case studies that evaluated the use of open source DES software in manufacturing systems. The first case study focused on a predictive capacity planner while the second case study focused on the performance of a semi-conductor manufacturing system. The authors noted the need for more holistic approaches to simulation

awareness/understanding, model build effort/time, and integration with existing systems as an area for future research. Heavey et al. (2014) proposed an open-source, cloud-enabled DES platform called DREAM to streamline the use of DES software packages. The DREAM platform consists of a simulation engine, a knowledge extraction tool, and a custom web-based graphical user interface (GUI). The platform was built using the Python programming language, and a Python library called ManPy was used to provide common simulation constructs within the Python environment. Two pilot case studies are used to elaborate on the extensibility of the tools in support of an industrial engineer in the context of scheduling and tool support. Future work will focus on incorporating new modeling elements and additional pilot case studies in different domains.

The complexity of DES software packages has also motivated interest in automated data-driven approaches that can reduce the amount of time and resources needed to create and conduct simulations. There are many challenges in developing an automated approach to simulation modeling, including incomplete data, dynamic/complex behavior (in the context of buffer and control strategies), and cyclic structures. Hybrid approaches, otherwise referred to as semi-automated approaches, have been proposed to address these challenges. Hybrid approaches help to reduce the time and expertise needed to develop complex simulation models by combining artificial intelligence with parametric methods that focus on the parameters of preexisting models, as well as structural approaches that focus on the structure of the model (Bergmann & Strassburger, 2010). Barlas, Dagkakis, &

Heavey (2015) introduced an automated knowledge extraction (KE) tool to gather data for a simulation model. The KE tool is an open source application created with different Python libraries (e.g., RPy2) to create an interface to the open source statistical software R within a Python environment. A case study was completed in a medical device fabrication facility to validate the KE tool. Results showed that there is an opportunity for savings on the total project time in the range of 10 to 40% due to the input data process. Future work plans to focus on expanding the simulation objects available and the creation of a GUI to provide a simpler solution. In a separate research study, Haraszko and Nemeth (2015) developed configurators (i.e., predefined templates addressing certain "species" of manufacturing systems) for the rapid creation of DES models for several types of manufacturing systems. A GUI was provided to guide a user through defining important characteristics of the system. Once the important process characteristics are defined, the DES model is automatically created and used for further optimization and improvement, which significantly accelerates the systematic design of manufacturing systems. The authors emphasized that there is a need for more affordable, easy-to-use modelling tools to support the rapid design of manufacturing systems.

Text-based approaches that create abstractions of manufacturing systems as a prelude to building simulation models help to isolate the user from the details of the simulation package. Gronniger, Krahn, Rumpe, Schindler & Volkel (2007) explained the many benefits that can be realized by users and developers when adopting text-based modeling methods. In particular, the authors described how

complex software systems (such as GUI-based simulation software packages) force users to focus on complex graphical representations that distract them from the core tasks at hand and limit the amount of information that can be represented on a single screen interface. In contrast, text-based approaches allow for high content density that permit a larger amount of information to be seen on a single interface, help users focus on core modeling tasks, and enable faster speed of creation. As a result, users are no longer concerned with the unique interfaces of a software system but, instead, focus on the structure of the model thus increasing the value of the modeling exercises. Similarly, Hughes, Scott, & Ridgway (2013) developed a DES module to support SMEs on automatically creating process flow DES models. This research demonstrated the feasibility of automatically generating a DES model based on input data from an online capture tool. Once the data have been captured from the input interface, a text-based representation of the model is created based on the user provided information and then transferred to the DES software system for execution, thereby reducing both the time to develop models manually from scratch and the required expertise.

## 2.3 DATA DRIVEN ANALYTICS AND SIMULATION

Data driven techniques incorporate concepts from data mining, process modeling, and analysis with the objective to discover, monitor, and improve processes by extracting knowledge from information systems (Aguirre, Parra, & Alvarado, 2013). The data are available as files composed of records of actions/steps that

occur within a process (referred to as *event logs*) and are typically used to enable the discovery of multiple perspectives about a process such as control-flow, performance, and resource information (Rozinat, Mans, Song, & W.M.P. van der Aalst, 2009).

Simulation models are often created manually using data collected via documentation, interviews, and close observation of the real-world process. This manual approach is time consuming and prone to errors due to the results being based on a subjective human understanding of the system. Researchers are beginning to recognize the benefits of merging simulation technologies and data-driven analytics (such as process mining) to gain better and faster insight into processes. For example, Rozinat et al. (2009) developed an approach to merge DES software and process mining with the objective of automating the simulation modeling process. Their approach was validated using two case studies in the Netherlands where the ProM process mining framework was used to discover multiple perspectives of an underlying process model such as control flow, data, performance, and resources. The process model discovered was then integrated into a comprehensive Colored Petri Net (CPN) simulation model that can be used for analysis. A control-flow discovery algorithm known as the *alpha algorithm* was used to create a process model automatically reflecting the causal relations among the activities captured in the event log. The automated support of redesign (i.e., suggesting process improvements based on log analysis and simulation

alternatives) as well as the use of simulation for real-time decision-making was noted as an area for future work.

Ahn, Dunston, Kandil, and Martinez (2015) conducted a study whose objective was to produce a DES model to abstract a process in a large earth moving operation using a refined version of the alpha algorithm. The main input was sensor-based time series data collected from construction equipment. The study focused on automatically generating the structure of a simulation model to reduce the large amounts of time required to produce an accurate model, and the dependence on expert knowledge and data pre-processing. Future work is planned for creating more practical algorithms related to process discovery and the creation of automated data-driven simulation models.

While many systems are very precise and controlled, manufacturing is rarely so controlled. Rozinat et al. (2009) investigated the applicability of data-driven analytics (i.e., process mining) in unstructured manufacturing processes at ASML, the world's leading manufacturer of chip-making equipment, to quickly identify bottlenecks and to develop ideas for improvement. This research study is unique in that case studies on process mining are typically based on structured administrative processes with correct *a priori* knowledge of the system. The ProM process mining framework was again used to perform process mining, as there are many pre-made plug-ins available for use. This study demonstrates that process mining techniques can yield concrete solutions for process improvement in complex environments such as the wafer scanner qualification phase of ASML.

More specifically, the results of testing one example machine showed that by filtering out problematic data and using process mining techniques, three of four testing phases that led to high amounts of idle time in the wafer scanner qualification phase could be identified. The authors noted that future research is needed to develop process mining techniques suitable for analyzing less structured processes.

## 2.4 LITERATURE REVIEW SUMMARY

Three clear research gaps have been identified through the review of the literature. The primary research gap calls for practical methodologies that aid in gaining a faster understanding of the behavior of manufacturing assembly systems and the rapid development of simulation models that relate to such systems. A second research gap calls for the development of data-driven methodologies for the modeling of manufacturing assembly systems of different complexities that contain dynamic resources such as finite capacity buffers, complex routing logic, and noise. Finally, the third research gap indicates that there is a need for simple, unified systems that can apply advanced analytical techniques without requiring significant domain expertise. If successfully developed, the aforementioned methodologies have the potential to automate the discovery of simulation models and their performance characteristics (e.g., reveal the correlation between buffer resources and overall throughput). However, the literature reviewed shows that these research areas are just beginning to be addressed.

Therefore, the objective of this research is to develop a methodology to automate the process of creating, simulating, and analyzing a manufacturing assembly system. The proposed methodology will address the need for simplified systems that can handle dynamic behavior and complex characteristics of manufacturing assembly systems such as finite capacity buffers and parallel workstations. To accomplish the research objective, knowledge from a variety of disciplines, including traditional process improvement, simulation, and process mining will be combined into one simple integrated solution. It is anticipated that by automating the simulation and analysis steps, a large portion of the project timeline can be bypassed to enable a user to focus more rapidly on value-added analysis tasks.

# 3 RESEARCH METHODOLOGY

This chapter introduces the proposed methodology to automate the modeling, simulation, and analysis of manufacturing assembly systems. The main phases of the proposed methodology are depicted in Figure 3.1.

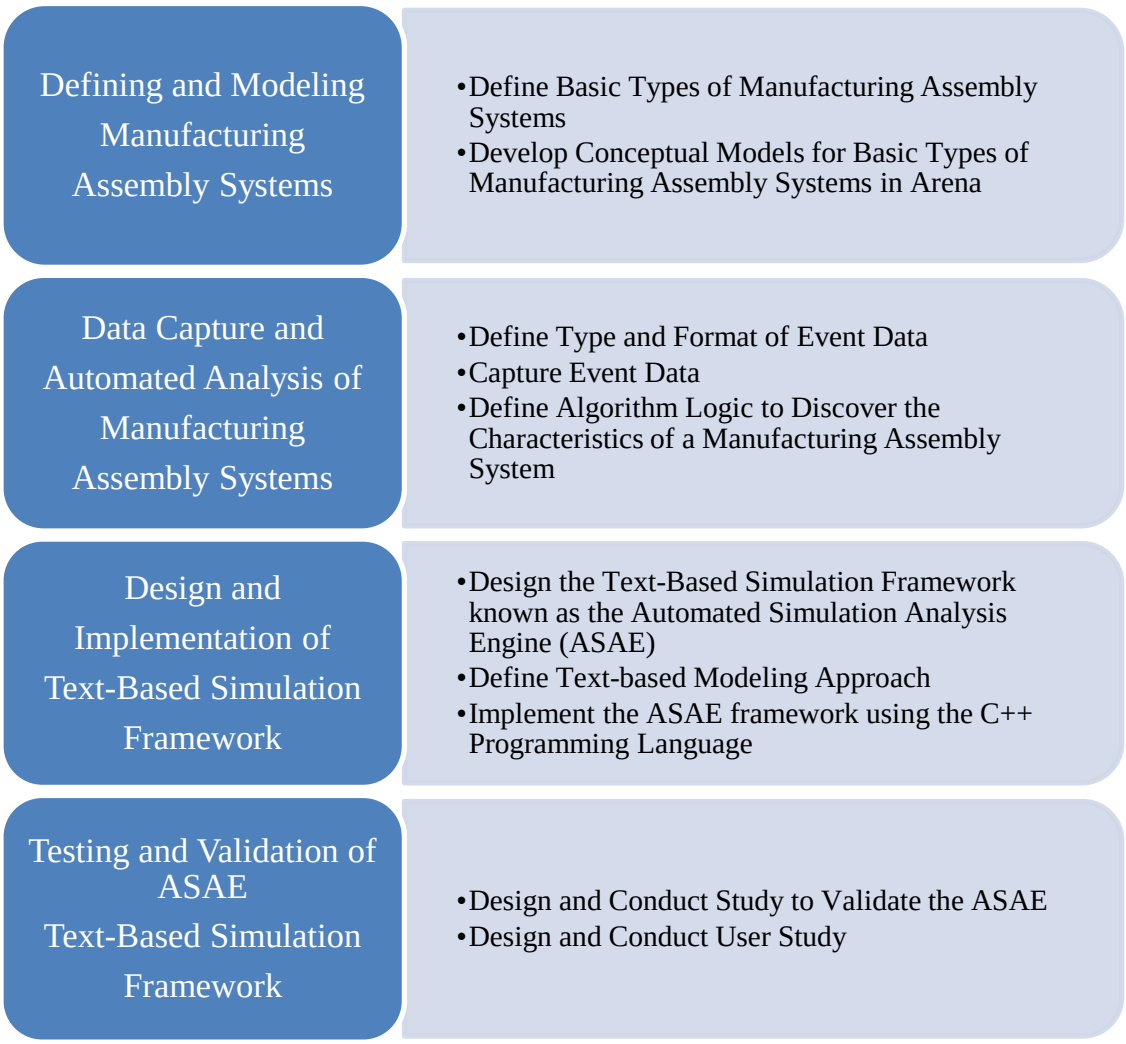



**Figure 3.1. Proposed Research Methodology.**

As depicted by Figure 3.1, the first phase of the methodology focused on the definition and modeling of several basic types of manufacturing assembly systems (MASs). The main objective when defining the basic types of MASs was to capture the essential features found in discrete manufacturing environments (e.g., product flow options, parallel processing, finite buffers, etc.). These basic types of MASs were then modeled and implemented using the discrete event simulation (DES) software Arena to better understand how data could be captured when events occurred during the simulated operation of these systems.

In the second phase of the methodology, the format of event data was defined as well as the required methods to extract event data from the simulated MASs to create an *event log*. The data in the event log were then used to discover process characteristics of the MASs such as process flow, throughput, buffer capacity, transition times, and process times since these process characteristics are not easily understood by simply building a model, but typically require specialized analysis to help understand the performance of the MAS. Algorithms were created that utilized the data in the event log to understand each of these process characteristics programmatically without the aid of a subject matter expert.

Phase three of the methodology focused on the design and implementation of a new simulation framework, which includes a text-based modeling approach that enables the automated simulation and analysis of MASs. The new simulation framework, called the Automated Simulation Analysis Engine (ASAE), was written in the C++ programming language. Finally, phase four of the methodology

focused on the design and execution of a validation study and a user study to assess the correctness and usability of the ASAE text-based simulation framework.

The rest of this chapter is organized as follows. Section 3.1 describes the steps taken to define the conceptual components of a MAS and how these elements are used within the modeling process. Section 3.2 explains the approach used to capture event data and how these data are used to automate analyses. Section 3.3 describes the process followed to design and implement the ASAE text-based simulation framework. Finally, Section 3.4 describes the approach to test and validate the correctness and usability of the ASAE text-based simulation framework.

## 3.1 Defining and Modeling Manufacturing Assembly Systems

The manufacturing sector is very competitive and encompasses a wide variety of activities including production management decisions and complex manufacturing engineering (Heavey & Robin, 2014). Similarly, the modeling and simulation of MASs is a complicated and time-consuming task that requires extensive expert knowledge that is critical in facilitating a better understand of the key characteristics of such systems (e.g., product flows, processing times, process dependencies, etc.) (Ahn, Dunston, Kandil, & Martinez, 2015). Fortunately, there are many methods and software solutions that can be used to model and understand MASs.

Given the large variety of MASs that could be modeled, this research limited its scope to three basic types to use as a basis to develop a new simulation approach and to keep the breadth of the research to a manageable size.

### 3.1.1 Define Basic Types of Manufacturing Assembly Systems

Three basic types of MASs were defined in this research to identify, understand, and test different modeling approaches. The three basic types of MASs are:

1. **MAS Type I.** A simple linear MAS.
2. **MAS Type II.** A MAS with parallel workstations and finite buffer resources.
3. **MAS Type III.** A MAS with parallel workstations, finite buffer resources, and probabilistic branching.

An example of a MAS Type I is depicted in Figure 3.2. This type of MAS is considered simple because it is composed of the most basic constructs of a process model (i.e., sequential processes with transitions).

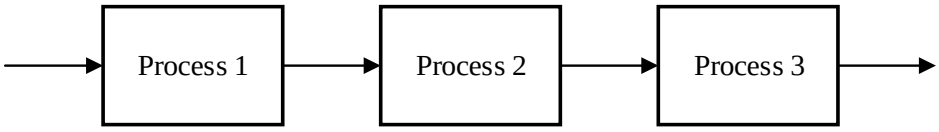



**Figure 3.2. Example of a MAS Type I.**

An example of a MAS Type II is depicted in Figure 3.3. This basic type of MAS was used to explore the process characteristics introduced into when jobs (i.e., *entities* in the context of DES), are processed in parallel, merged, and possibly stored temporarily in finite capacity buffers.

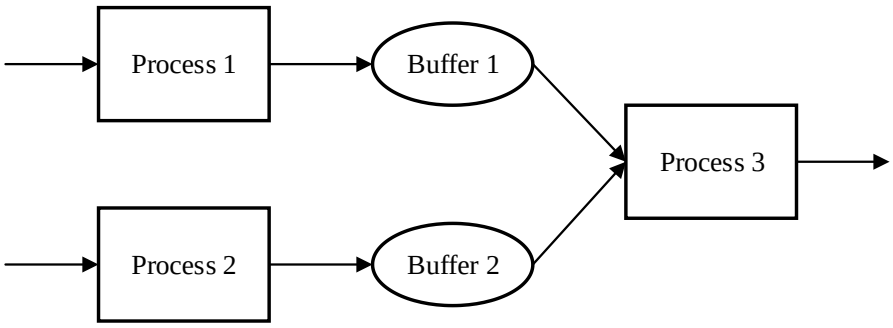



**Figure 3.3. Example of a MAS Type II.**

Figure 3.4 depicts an example of a MAS Type III which incorporates parallel workstations, finite buffer resources, and probabilistic branching (e.g.,

rework and/or defects). The MAS Type III was constructed to represent more realistic systems. The modeling knowledge developed in constructing the previous (and simpler) models in Arena facilitated the understanding of more complex MASs that closely resemble reality.

The following section describes the details of modeling the three basic types of MASs using Arena.

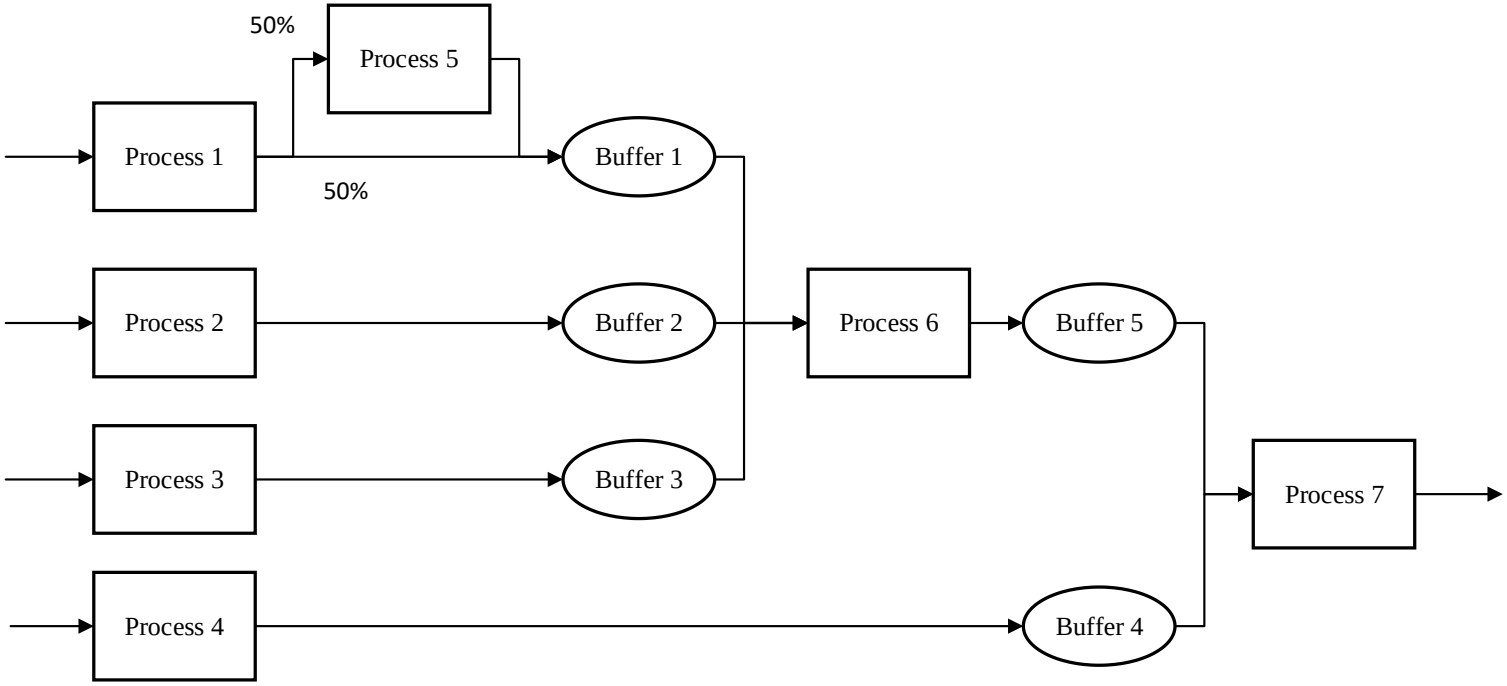



**Figure 3.4. Example of MAS Type III.**

### 3.1.2 Develop Conceptual Models for Basic Types of Manufacturing Assembly Systems in Arena

This research leveraged the capabilities of Arena, a DES simulation software that is widely used in industry and educational settings, to understand the process of modeling the three basic types of MASs introduced in Section 3.1.1. One of the key features of Arena is its graphical user interface (GUI), which facilitates the definition and creation of simulation models. Through this GUI, a variety of predefined components and/or modules can be dragged into Arena's simulation panel to fulfill the typical functions found in a MAS.

Most commercially available DES simulation software packages use the concept of an *entity* to represent a job moving through a MAS (Kelton, Sadowski, and Zupick, 2010). Other components available in Arena that are used to represent additional constructs of a MAS include *resource*, *process*, and *variable*. Table 3-1 explains in more detail the purpose of these components.

**Table 3-1. Components Available in Arena to Build Simulation Models.**

| Component Type | Description |
|---|---|
| **Entity** | Represents an individual instance of a job. |
| **Resource** | User defined element that can be used to define workers, buffers, materials, etc. |
| **Process** | Represents a specific step within a simulation model. |
| **Variable** | Defined values that can be tracked and updated to reflect characteristics of a process. |

Arena also provides a variety of modules to control the simulation of a model. The modules available within Arena that were used to create the simulation models of the three types of MASs are listed and described in Table 3-2. Arena offers additional modules that can be used to model many types of systems, but not all the modules were relevant for the purposes of this research.

**Table 3-2. Modules Available in Arena to Build Simulation Models.**

| Component Type | Description |
|---|---|
| **Create** | Used to create entities and to define inter-arrival times and the number of entities per arrival. Represents an entry point into a DES model. |
| **Assign** | Used to assign a value to a pre-defined variable at a certain location in the DES model when an entity arrives to the module. |
| **Seize** | Used to seize a defined resource within the DES model. |
| **Delay** | Used to represent a time-oriented event. When an entity hits a delay module, the entity waits a defined period of time before moving forward through the DES model. |
| **Process** | Used to group Seize, Delay, and Release modules into a single module. The Process module seizes a defined resource, delays it for a defined period of time, and releases the resource for further use. |
| **Hold** | Used to hold an entity until a defined condition is met. |
| **Decide** | Used to define branches within the DES model based on pre-defined probabilities or percentages. |
| **Release** | Used to release a defined resource when triggered by the arrival of an entity. |
| **VBA** | Used to develop custom modules that execute user-defined code that is triggered by the arrival of an entity. |
| **Match** | Used to match a defined number of entities before releasing them further into the DES model. |
| **Batch** | Used to merge multiple entities into a single entity. |
| **Dispose** | Used to remove entities from the DES model. |

Depending on the basic type of MAS to be modeled, a user must select the appropriate combination of components and modules available in Arena and connect them to represent a real-world MAS as closely as possible.

#### *3.1.2.1 Modeling the Simple Linear MAS in Arena*

The simulation model of a conceptual simple linear MAS (i.e., MAS Type I) can be built in Arena by linking several Process modules in a sequence. Alternatively, a MAS Type I can be built by using a set of Seize, Delay, and Release modules linked together in a sequence. Once the MAS Type I is built using one of these options (or a combination of the two), entities (i.e., jobs) will enter the MAS through a single Create module and will exit the system through a single Dispose module. Prior to running the simulation, the Seize and Release modules must be configured to utilize available resources that have been defined within Arena.

In this research, the MAS Type I was composed of three processes, each with separate Seize, Delay, and Release modules. The complete Arena simulation model of the MAS Type I is depicted in Figure 3.5. Separate Seize, Delay, and Release modules were used to provide clarity in each step of the modeling process, to allow for the addition of custom logic, and to facilitate the recording of event data at distinct points in the simulation model. The separate modules facilitate a finer level of control when specific resources are seized and when event data are recorded into comma-separated value (CSV) files.

As Figure 3.5 shows, the Arena simulation model of the MAS Type I begins with a Create module that defines the entry point into the MAS. The Create module is followed by three sequential process steps, each represented by a combination of

Seize, Delay, and Release modules. The Arena simulation model is completed with a single Dispose module to serve as an exit from the MAS.

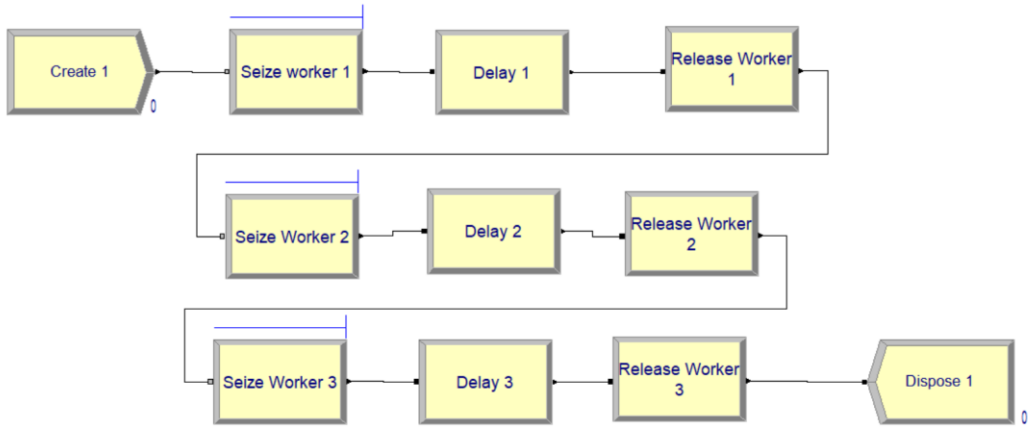



**Figure 3.5. Simulation Model of a MAS Type I in Arena.**

### *3.1.2.2 Modeling a MAS with Parallel Workstations and Finite Buffer Resources in Arena*

The addition of parallel workstations and finite buffer resources rapidly increases the complexity of creating a simulation model in Arena. Therefore, a MAS Type II was created in Arena to understand the main modeling challenges, and to learn how to overcome these challenges in future DES software. It is important to note that Arena does not have a pre-defined module that can be used to represent finite buffer resources in a MAS. Therefore, custom logic must be developed to add this behavior to a simulation model.

There are many approaches to model a finite buffer resource in Arena. In this research, finite buffer resources were defined with a set capacity (i.e., an available number of buffer slots). A buffer slot in the finite buffer resource is seized when an entity exits a process and released when the entity enters the next process. If a finite buffer resource has no capacity left, then an entity will have to wait until a buffer slot is available.

The logic to implement this specific modeling approach must be interleaved with the process resources so that as an entity flows through the MAS, the finite buffer resource is controlled correctly when processes begin and finish, respectively. More specifically, a process cannot be made available (i.e., released in Arena) until an available buffer slot in the finite buffer resource can be seized, which means that an entity has moved out of the process module and into a buffer slot. Making the logic required to represent a finite buffer resource "transparent" to a user is one characteristic this research aims to simplify.

Figure 3.6 depicts the Arena simulation model of a MAS Type II. The model starts with two Create modules that are used to initiate the two branches of the MAS. On each branch, the Create modules are followed by a sequence of Seize, Delay, and Release modules. Since a MAS Type II contains finite buffer resources, the modules labeled "Release worker 0" and "Release worker 1" take place immediately after their corresponding buffer modules are seized (i.e., "Seize Buffer 0" and "Seize Buffer 1") and not directly after the Delay modules (i.e., "Delay 0" and "Delay 1"). Similarly, the resources labeled "Seize Buffer 0" and "Seize Buffer

1” are not released until the next worker resource is able to be seized (i.e., module labeled “Seize worker 2”). Before the next worker is seized, the individual entities coming from the two branches must be merged into a single entity. The merging of entities is accomplished through the Match and Batch modules labeled “Assembly Match” and “Batch 1”, respectively. The Match module ensures that there is a matching pair of entities and the Batch module takes the two entities and merges them into a single entity. Once merged into a single entity, the next worker can be seized, and the buffer resources released, which is represented with the modules labeled “Seize worker 2” and “Release Buffers 0 and 1”, respectively. This pattern of seizing a buffer resource and later releasing it simulates the use of finite buffer resources within the Arena DES software. The entities exit the MAS via a single Dispose module.

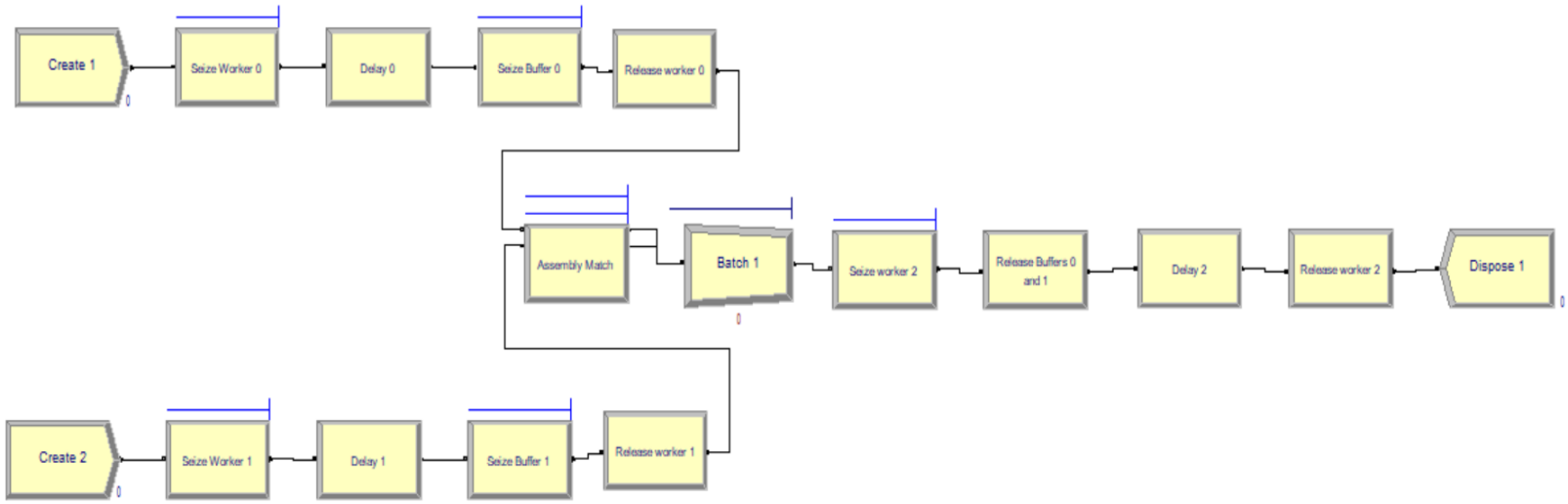



**Figure 3.6. Simulation Model of a MAS Type II in Arena.**

#### *3.1.2.3 Modeling a MAS with Parallel Workstations, Finite Buffer Resources, and Probabilistic Branching in Arena*

Since the approach to model parallel workstations that merge into another workstation and finite buffer resources in Arena has already been discussed in Section 3.1.2.2, the focus of this section will be on describing how to model probabilistic branching. This modeling feature is used in Arena to incorporate the randomness and inconsistences of real MASs.

The module Decide is used in Arena to branch a process based on a set of user-defined probabilities. For example, a MAS may have two processing branches each taken 50% of the time. In this case, the Decide module would be connected to two paths with conditional probabilities set to 0.5, respectively. Processing errors, which are often represented as a probabilistic branch, are common in real MASs and are important characteristics to capture in a simulation model. The ability to model probabilistic branching was one particular characteristic this research wanted to incorporate into the proposed ASAE text-based simulation framework.

Figure 3.7 depicts the complete Arena model of a MAS Type III. While the details of the modules cannot be read, Figure 3.7 is used to show general structure and complexity. There is a single Decide module in this Arena simulation model (see the rhomboidal icon at the end of the first processing branch), which represents the inclusion of rework into the DES model.

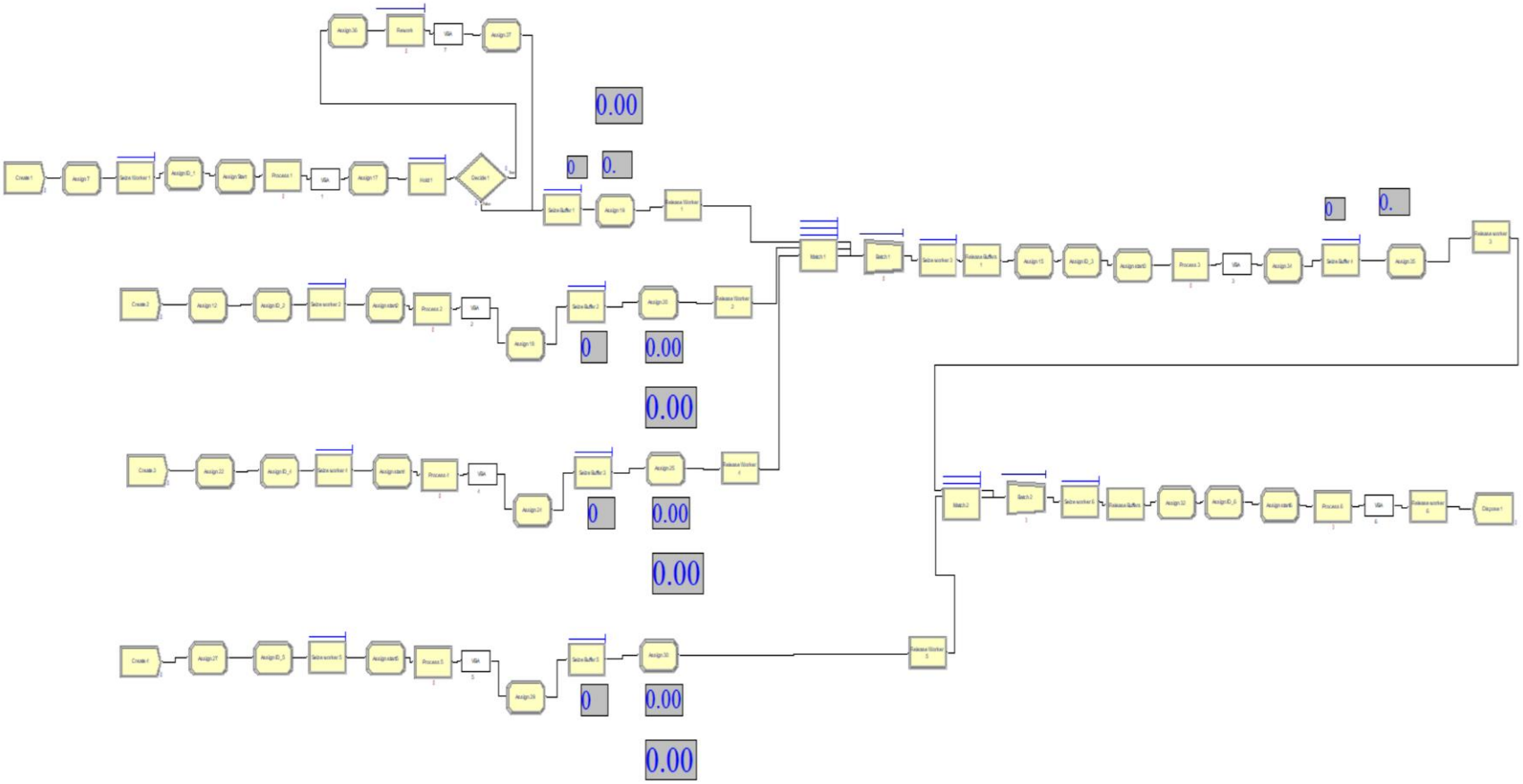



**Figure 3.7. Simulation Model of a MAS Type III in Arena.**

## 3.2 Data Capture and Automated Analysis of Manufacturing Assembly Systems

The Arena simulation models of the three basic types of MASs were used to explore:

- How to extract data from the simulation model of a basic type of MAS,
- How to define the format of these data to enable effective analysis, and
- How to leverage these data programmatically to automate the analysis of a basic type of a MAS.

Each of these focus points were explored within the context of Arena and were used to build an initial understanding of how to handle event data in a DES software package. Collecting formatted event data from a simulation model enables the programmatic analysis of MASs without the need of a simulation subject matter expert (SME).

### 3.2.1 Define Type and Format of Event Data

When executing the simulation of a MAS using DES software, specific event data are often collected to gain insight into its performance. Examples of these event data may include start time, end time, and the number of jobs processed, to name a few. Furthermore, the event data should be collected in a specific format to enable certain types of analyses.

When selecting the format of event data to be captured, it is important that the details chosen are not too high-level as to not capture the intricacies of the MAS

but, instead, be organized and recorded at a level that enables individual jobs to be identified at each process step. To enable this, event data such as the “start time” and “end times” for each process are recorded with an associated “job ID” and “process ID” to provide a distinguishable record that can be linked to a particular job instance at a certain process within a simulation model.

It is also crucial that there is a method for identifying process flow information within the simulation. In this research, process flow information was discovered through the field “process ID”, which is recorded for every start event. The field “process ID” for start events is a composite ID that combines the identification numbers of the previous process (listed first) followed by the current process. For example, if Process A is the first in a MAS followed by Process B, then the start records associated with Process A would be identified by “A”, because there are no jobs before it. However, the start events for Process B would be identified with the composite ID “AB” to indicate that the job moved from process A to process B. Recording process flow in this manner allows for an instance of a job to be tracked through the MAS and provides insight into process flow. The process ID format just described is illustrated in the column labeled “ProcessID Enter” in Table 3-3, which also shows additional columns that track the job instance, start and finish times, and resource information. The event data enables the analysis of process flow, throughput, and buffer capacity.

**Table 3-3. Example Event Data Extracted from a Simulation in Arena.**

| JobID Enter | ProcessID Enter | StartTime | Resource | JobID Exit | ProcessID Exit | EndTime |
|---|---|---|---|---|---|---|
| 1 | A | 0 | Worker 1 | 1 | A | 0.506561 |
| 2 | A | 0.506561 | Worker 1 | 2 | A | 1.177419 |
| 3 | A | 1.177419 | Worker 1 | 3 | A | 1.712956 |
| 1 | AB | 0.506561 | Worker 2 | 1 | B | 1.735751 |
| 4 | A | 1.712956 | Worker 1 | 4 | A | 2.400493 |
| 5 | A | 2.400493 | Worker 1 | 5 | A | 2.876961 |
| 2 | AB | 1.735751 | Worker 2 | 2 | B | 3.199425 |
| 6 | A | 2.876961 | Worker 1 | 6 | A | 3.495455 |
| 7 | A | 3.495455 | Worker 1 | 7 | A | 4.093871 |
| 3 | AB | 3.199425 | Worker 2 | 3 | B | 4.429114 |
| 8 | A | 4.093871 | Worker 1 | 8 | A | 4.595768 |
| 9 | A | 4.595768 | Worker 1 | 9 | A | 5.155075 |
| 4 | AB | 4.429114 | Worker 2 | 4 | B | 5.679988 |
| 10 | A | 5.155075 | Worker 1 | 10 | A | 5.771469 |

### 3.2.2 Capture Event Data

To record and extract event data while a basic type of MAS was simulated, custom code was written in Arena's Visual Basic for Applications (VBA) programming language and integrated into the simulation model via the VBA module. The custom VBA code was written to automate the process of collecting event data (i.e., start times, end times, process ID, and job ID) and to write these data into a Microsoft® (MS) Excel spreadsheet for use in analysis.

The custom VBA code added to Arena can be setup to run at various points within the simulation (e.g., when the simulation starts, when the simulation ends, or at the arrival of an entity to a module). For the purposes of this research, the

VBA code used to extract event data from the simulation of a MAS was executed when an entity entered the VBA module.

Figure 3.8 illustrates an example of the VBA code written for a function (i.e., VBA_Block_3_Fire()) that is used in Arena to collect event data for each entity that enters a VBA module. At the beginning of the function, a few variables are first defined to store values from Arena's Application Programming Interface (API). The simulation is then queried to obtain the values to be stored in these variables. Next, variable values are stored in an MS Excel worksheet. Finally, the row number is incremented for the next event.

The VBA code structure shown in Figure 3.8 was repeated at various locations in an Arena simulation model to extract event data at each process step. The complete listing of VBA code integrated into the Arena simulation model of each basic type of MAS is included in Appendices A, B and C.

```
'Function to record event data when an entity enters the VBA module
Private Sub VBA_Block_3_Fire()

    'Define Variables
    Dim jid As Long
    Dim simTime As Double, startTime As Double

    'Assign variables values from Arena's Application Programming
    Interface (API)
    startTime = oSiman.EntityAttribute(oSiman.ActiveEntity,
    startTimeindex3)
    simTime = oSiman.RunCurrentTime
    jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

    'Place values into excel worksheet
    With oWorksheet
       .Cells(nNextRow, 1).value = jid & "C"
       .Cells(nNextRow, 2).value = "Process 3"
       .Cells(nNextRow, 3).value = startTime
       .Cells(nNextRow, 4).value = simTime
       .Cells(nNextRow, 5).value = "Worker 3"
    End With

    'Increment to the next row in the excel worksheet
    nNextRow = nNextRow + 1
End Sub
```

**Figure 3.8. Example VBA Code for a Custom VBA Module in Arena.**

### 3.2.3 Define Algorithm Logic to Discover the Characteristics of a Manufacturing Assembly System

The analysis of a MAS can be challenging if insufficient event data are captured during the simulation. However, as described in Section 3.2.1, when event data is recorded with distinct job IDs, process IDs, and start times, characteristics of the MAS such as process flow, process times, and transition times can be discovered. Discovering process characteristics would not be possible if DES software packages just simulated the model of a system and terminated. Instead, DES software packages simulate and track the performance characteristics of a system to provide insight into a model.

In this research, the event data extracted from the Arena simulation models of the different types of MASs were used to drive the development of algorithms to help understand process characteristics programmatically including process flow, throughput, transition frequencies, transition times, utilized buffer capacity, and process performance.

#### *3.2.3.1 Unique Processes*

One of the main uses of event data collected from the MASs simulated in Arena was to distinguish individual processes within a MAS. The identification of individual processes was accomplished by iterating though each event record for a finishing process ID (i.e., “JobID Exit” in the MS Excel sheet) and constructing a

set of process IDs. After iterating through every entry of the event log, all the unique processes within the MAS are known.

#### *3.2.3.2 State Transitions*

The use of event data to better understand how jobs flow through a MAS can be challenging. This research addressed this challenge through the use of a transition state matrix (TSM). In the context of a MAS, a TSM represents the connections that exist between unique processes. Once the initial TSM of a MAS is constructed, valid transitions (i.e., paths that entities may take from one process to another) can be identified. In addition, once the TSM is populated with frequency information, analyses can be performed to understand process flow. Table 3-4 shows an example of how a TSM can be used to represent process flow.

**Table 3-4. Example of a Transition State Matrix.**

| **PID** | **A** | **B** | **C** |
|---|---|---|---|
| **A** | A to A | A to B | A to C |
| **B** | B to A | B to B | B to C |
| **C** | C to A | C to B | C to C |

Programmatically, a TSM is created in the C++ programming language using a two-dimensional array of size *M* by *M*, where *M* represents the number of unique processes and each index represents a transition. The counts for each process

transition (i.e., represented with an index in the two-dimensional array) are determined by iterating through the event data in the TSM.

The TSM can also be used to understand flow through a system via the dependency information contained within the process ID (i.e., see the columns labeled “A”, “B”, and “C” in Table 3-4).

#### *3.2.3.3 Terminal States*

The simulation model of a MAS contains both entry points (i.e., points at which entities enter the MAS) and exit points (i.e., points at which entities leave the MAS). When determining the throughput of a MAS, it is necessary to know how many exits a MAS has. In the context of a TSM, the exit of a MAS is referred to as a *terminal state*, which means that an entity is terminated when it reaches that position. Once a TSM has been constructed, the identification of all terminal states is a trivial task since a terminal state will have zero connections.

Terminal states are represented as rows that contain zero transitions in a TSM, as illustrated by the row for PID C in the TSM shown in Table 3-5. When each row with zero transitions is identified, each exit from a system (i.e., terminal state) is known and the throughput characteristics can be further understood.

**Table 3-5. Example of a TSM with PID C Representing a Terminal State.**

| PID | A | B | C |
|---|---|---|---|
| **A** | 0 | 100 | 0 |
| **B** | 0 | 0 | 100 |
| **C** | 0 | 0 | 0 |

#### *3.2.3.4 Jobs Completed and Throughput*

The jobs completed per time unit, also known as *throughput*, is an important metric that helps in understanding how well a MAS is performing. To determine the throughput of a MAS, all terminal states (i.e., process flow information) must be known ahead of time to understand when entities are exiting the MAS. Once the terminal states are known (i.e., rows of a TSM with zero transitions), the event data can be processed to determine how many jobs are completed at each terminal state.

The throughput can be described for each individual terminal state or for the entire MAS by providing the following two metrics:

1. The count of all jobs exiting the MAS at a position or from the entire MAS, and
2. The time per job based on the simulation time.

The metric "time per job" is calculated by dividing the total simulation time by the number of jobs completed. Calculating the "time per job" for a terminal state or for the entire MAS involves the same calculation.

#### *3.2.3.5 Buffer Identification and Maximum Utilized Capacity*

Understanding how dynamic resources (i.e., finite buffers) are being utilized within a MAS can be challenging. Therefore, a performance metric to reflect the maximum capacity utilized by a given buffer resource at a valid transition was calculated. Figure 3.9 depicts the process to calculate this performance metric, which consists of the following steps:

1. **Find Valid Transitions.** Collect and organize all valid job IDs from the upstream process that will be passed to the downstream process. This collection can be seen as set T in Step 1 of Figure 3.9. This step is required since probabilistic branching is possible at any transition, and only the jobs that are valid for a given transition should be used.
2. **Find Finite Buffer.** Determine if a buffer is present by using a "sliding window" approach. In the "sliding window" approach, the window is the gap between sequential jobs. Three sequential job instances within a connection (i.e., Jobs 1, 3, and 4 in Figure 3.9) are extracted from the data obtained in Step 1 to check if a job (i.e., Job 3) is in the window. If a job is in fact stuck between two running processes, this situation indicates the presence of a buffer.
3. **Find Maximum Finite Buffer Capacity.** The maximum utilized capacity of the finite buffer is then calculated by expanding and shrinking the "sliding window" as jobs start and finish. As shown in

Step 2 of Figure 3.9, if the downstream process is working on job ID 1 and the upstream process is working on job ID 4, then Job 3 is in the buffer. As seen in Step 3, if the upstream process finishes two more job and starts Job 8, then Jobs 3, 4, and 7 are all in the buffer (i.e., the "sliding window" is expanded). Similarly, if the downstream process finishes Job 1, then one job has exited the buffer thus shrinking the "sliding window". This logic continues for all event data and, each time a new maximum capacity is discovered, the value is updated. In the end, the value of the maximum capacity of a buffer is returned and recorded for that transition.

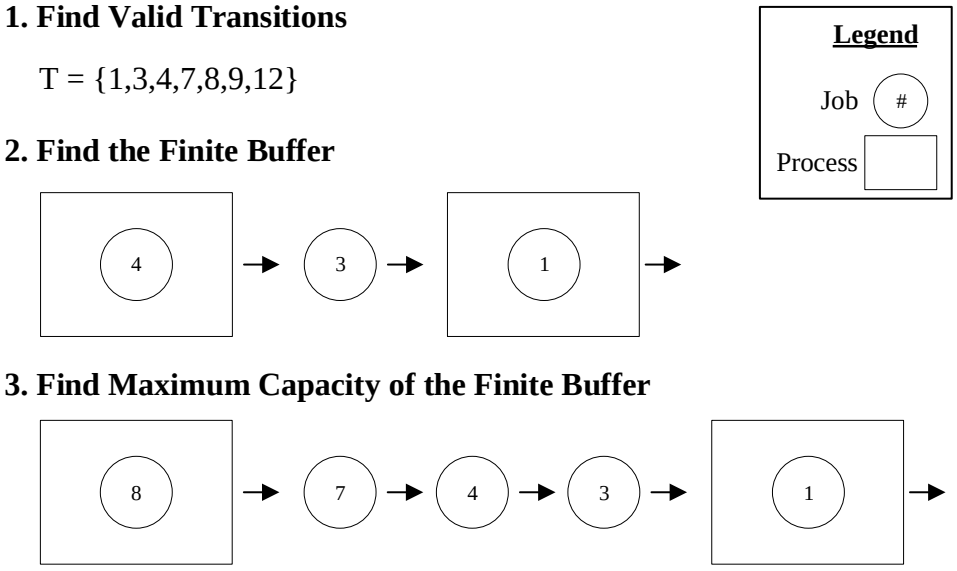



**Figure 3.9. Steps to Calculate Maximum Utilized Buffer Capacity.**

#### *3.2.3.6 Process Performance*

The process time is important when trying to understand characteristics such as throughput or when trying to identify the presence of bottlenecks. To provide insight into the performance of a process, the average process time is provided. This metric is determined by finding the start and finish record for a job instance at a given process and comparing the difference in simulation time. This is then repeated for each start and finish record and summed into a total process time. When each process time has been aggregated, the total is then divided by the number of jobs completed at that process to provide the average process time.

## 3.3 Design and Implementation of the Text-Based Simulation Framework

The knowledge gained after modeling, simulating, and analyzing the basic types of MASs in Arena was applied to the design of a new text-based simulation framework known as the Auto Simulation Analysis Engine (ASAE). The ASAE text-based simulation framework seeks to simplify the process of defining, building, and executing the simulation of a MAS and is comprised of the following three primary components (also depicted in Figure 3.10):

1. A text-based modeling approach,
2. A simulation engine, and
3. Automated analysis of process characteristics.

The ASAE text-based simulation framework allows a user to abstract a MAS process into simple components that can be passed to the simulation engine. The simulation engine can execute and analyze a simulation with just a text-based description of the MAS provided by the user, thus simplifying the simulation task and reducing the amount of experience needed to understand the simulation. The automated analysis allows the user to quickly acquire an understanding of the process characteristics without having to sort through process data manually thus saving time and resources.

### 3.3.1 Design of the Automated Simulation Analysis Engine Text-Based Simulation Framework

Simulation software packages are complex systems composed of many different elements including time, events, entities, resources, processes, and buffers, to name a few. When combined, these elements mimic the behavior of a real-world MAS.

The process of building a simulation model requires a great deal of time, as well as a deep understanding of not only simulation concepts but also the simulation software being used. In the case of the Arena simulation software, the user must understand how to create a simulation model using its GUI-based interface and how to define each module. Upon creating an initial simulation model, the specific parameters of the simulation run must be defined. Once the simulation run parameters have been defined and set, the simulation model can be executed. After the simulation ends, a base set of performance metrics based on entities, queues, and resources is presented. Each one of the steps just described requires knowledge of the Arena simulation software as well as simulation concepts.

The main objective in designing the ASAE text-based simulation framework was to eliminate the steep learning curve experienced by users when creating, running, and analyzing a simulation model with most of today's commercially available simulation software packages. The ASAE text-based simulation framework accomplishes this goal by simplifying the number of steps required to complete a simulation study.

In the ASAE text-based simulation framework, a simple text-based model file describing the structure of the MAS is passed to a simulation engine. The simulation engine then constructs a model of the MAS, executes the simulation, collects process data, and performs the initial analysis. The simulation engine component of the ASAE text-based simulation framework was created with the C++ programming language, and can be executed on any platform capable of running C++ programs. The simplified approach used by the ASAE text-based simulation framework completes the core simulation tasks, which include model construction, model execution, data collection, and analysis, as depicted in Figure 3.10.

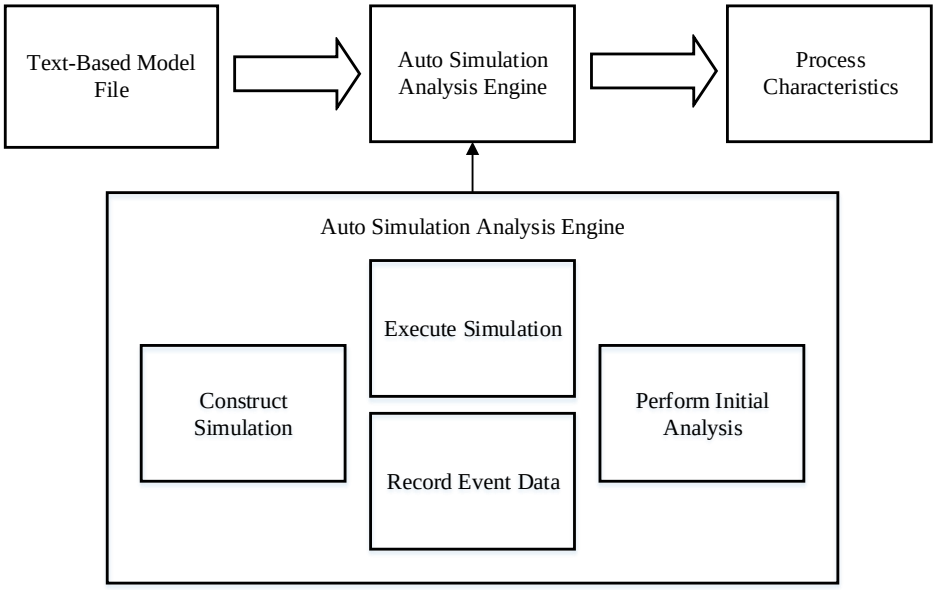
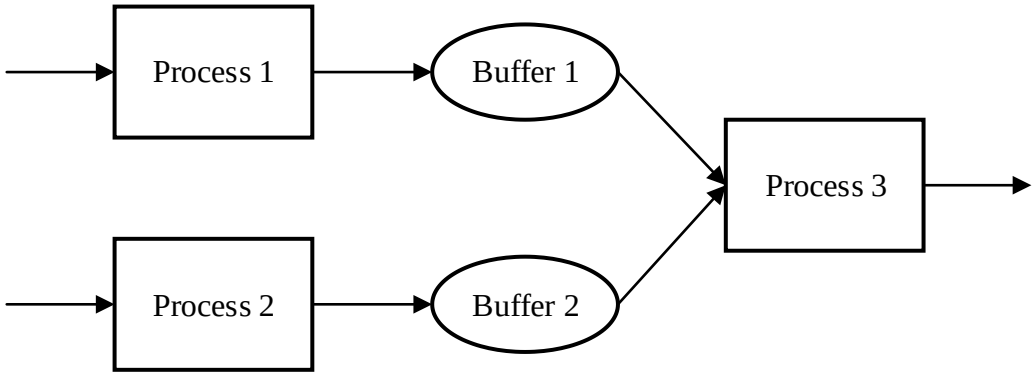



**Figure 3.10. Main Components of the Auto Simulation Analysis Engine.**

The creation of a proper interface to describe the structure of the MAS was one significant challenge faced in the design of the ASAE text-based simulation framework. Arena uses a GUI but, in the case of the ASAE simulation framework, a text-based approach was used instead. The text-based interface allowed for the core details of the MAS to be defined in one location and in a simple, easily understood format.

### 3.3.2 Definition of the Text-based Modeling Approach

In the ASAE text-based simulation framework, a text file is used by the user to describe what the simulation engine should construct and simulate. The text file must contain data about the simulation model including process characteristics, process flows, finite buffer characteristics, and the terminating condition (i.e., the number of jobs simulated). The text file is the only component of the simulation the user needs to understand. By separating the model definition task from the core simulation system task, the user does not have to be concerned with the intricacies of constructing and executing a simulation, or even the collection and analysis of data. An example of a complete text file for a MAS with three processes and two finite buffers is depicted in Figure 3.11. The line numbers shown in Figure 3.11 are not part of the actual text file used in the ASAE text-based simulation framework, but are included here to facilitate the explanation.




```
1:   <MODEL>
2:   <100>
3:   <3>
4:   <PROCESS 1>
5:   <T:3:4:5>
6:   <0>
7:   <1,03(1.00)05>
8:   <0>
9:   </PROCESS 1>
10:
11: <PROCESS 2>
12: <T:5:6:7>
13: <0>
14: <1,03(1.00)05>
15: <0>
16: </PROCESS 2>
17:
18: <PROCESS 3>
19: <T:2:3:5>
20: <2>
21: <0>
22: <2,(01,0),(02,0)>
23: </PROCESS 3>
24: </MODEL>
```

**Figure 3.11. Example of a Text File used in the ASAE Simulation Framework.**

Each process characteristic must be entered in the text file in a concise and organized format similar to that of the hypertext markup language (HTML) or the extended markup language (XML). As shown in Figure 3.11, the entire simulation model is contained within a model tag with the format <MODEL> </ MODEL >. The simulation model tag contains the number of jobs to simulate (line 2), the number of process steps (line 3), and a process block for each process (lines 4, 11, and 18).

Similarly, each process block is contained within a numbered process tag with the format <PROCESS #> </ PROCESS #>. The process tag contains four parameters:

1. Process Time.
2. Process Type.
3. Downstream Connections.
4. Upstream Connections.

The first line in the <PROCESS #> block (lines 5, 12, and 19) describes the statistical distribution (and its corresponding parameters) that most closely characterizes the process time. The ASAE text-based simulation framework allows a user to employ four different statistical distributions to characterize process times. Table 3-6 list these statistical distributions and shows the format that must be followed to define them in the text file.

**Table 3-6. Statistical Distributions used by the ASAE Text-Based Simulation Framework to Describe Process Times**

| Statistical Distribution | Text File Format |
|---|---|
| Triangular | T:Low:Avg:Upper |
| Normal | N:Average:stdDev |
| Uniform | U:Lower:Upper |
| Constant | C:Value |

The next line within the process block defines the position type (lines 6, 13, and 20). The position type parameter is used to describe (1) the position of the process (relative to other processes in the MAS), and (2) whether there are jobs before or after the process. The ASAE text-based simulation framework will treat jobs at processes differently depending on whether the job is at the beginning, middle, or end. Three options can be entered in the text file to indicate whether a process is at the beginning (i.e., 0), in the middle with upstream and downstream connections (i.e., 1), or at the end indicating a terminal position (i.e., 2).

The third line within the process block (lines 7, 14, and 21) describes the downstream connections of a process, which tells the ASAE text-based simulation framework how to handle jobs after they are completed (i.e., describes where to place the job when completed and what buffer the job will be placed in). The third line in the process block is also used to describe the probability of taking a given downstream process connection. In many simulation models, probabilities are used to control how jobs are passed through the system when there is more than one

branch that can be taken at a point in the MAS. Every process is also associated with a buffer with a certain capacity. This format is illustrated in line 7, where PROCESS 1 has only one downstream connection going to process 3. Since there is only one possible connection, the probability is expressed as 1.00 (i.e., 100%). The associated buffer (i.e., Buffer 1) has a capacity of 5. This information is entered into the text file as **1,03(1.00)05**. Another example for a process that has two downstream connections to processes 3 and 4 with a 50% probability for each branch, and buffer capacities of 5 and 5, respectively, is entered into the text file as **2,03(0.50)05,04(0.50)05**. In the case that a process has no downstream connections (i.e., a terminal position), the third line of the process block is filled with a <0>, as is the case with PROCESS 3 in the example depicted in Figure 3.11.

Finally, line four within the process block (lines 8, 15, and 22) indicates the upstream connections of a process. The values of the parameters specified in this line of the text file tell the ASAE text-based simulation framework where a received job is coming from. More specifically, the parameter values define from which process and from which buffer to pull the next job when it is completed at a process. In the example text file depicted in Figure 3.11, line 8 and line 11 show a <0> because PROCESS 1 and PROCESS 2 do not have an upstream process. However, PROCESS 3 (see line 22) has both PROCESS 1 and PROCESS 2 as upstream connections. PROCESS 1 and PROCESS 2 both have independent buffers to store their completed jobs, and thus when PROCESS 3 is ready to start a job, one job must be pulled from PROCESS 1's buffer and one job must be pulled from

PROCESS 2's buffer. Since both PROCESS 1 and PROCESS 2 only have one buffer, the line in the text file will indicate to pull from the first available buffer identified with an index of 0. The line to represent this process logic is then entered in the text file as **2,(01,0),(02,0)**.

#### *3.3.2.1 Web Interface to Create a Text-File for a Model*

The details for creating the text file used by the ASAE simulation framework can be difficult to remember. Therefore, a web-based tool was developed to aid the user in the creation of the text file needed to simulate a model.

The web-based interface, depicted in Figure 3.12, was implemented using HTML, JavaScript, cascading style sheets (CSS), and the React framework. The web-based interface provides a structured template for the data entry process and is composed of four main panels:

- Model Definition
- Process
- Format
- Model File

The Model Definition panel (see Figure 3.12, label 1) allows the user to initialize the structure of a simulation model via two input fields: "Number of Jobs" and "Number of Processes". The value entered in the input field "Number of Jobs" defines how many jobs will be simulated in the model, whereas the value entered

in the field “Number of Processes” specifies how many process steps must be defined by the user.

After values for the number of jobs and the number of processes are entered, the user must press the “DEFINE” button to enable the Process Panel (see Figure 3.12, label 2). The Process panel is used to define the parameters of each process found within a MAS, including “Process Time”, “Position Type”, “DownStreamConnections”, and “UpstreamConnections”. After defining the parameters for each process block, the “ADD” button must be pressed to update the model file.

The Format panel (see Figure 3.12, label 3) displays examples of how each process parameter must be formatted when entered in the Process panel. Finally, the Model File panel (see Figure 3.12, label 4) displays the model file being created and allows the user to edit the information already entered, if needed.

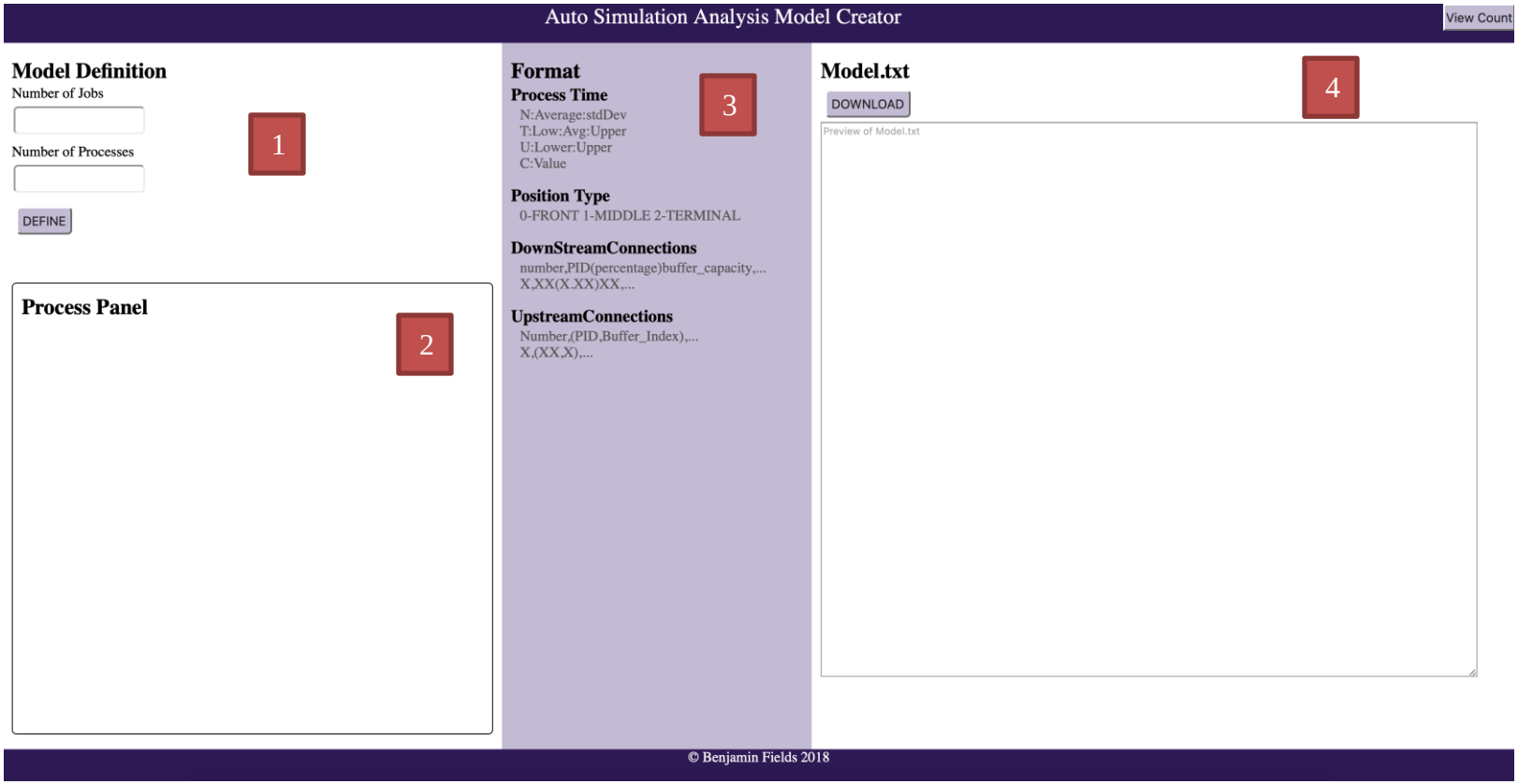



**Figure 3.12. Auto Simulation Analysis Model Creator Online Interface.**

### 3.3.3 Implementation of the Simulation Engine of the ASAE Text-Based Simulation Framework

The simulation engine of the ASAE text-based simulation framework was implemented using the C++ object-oriented programming language. The simulation engine is composed of five primary classes, as shown in Table 3-7.

**Table 3-7. Classes of the Simulation Engine**

| Class Name | Description |
|---|---|
| **Simulation** | Main controller of the simulation. Used to create and run a simulation model. Contains the logic for constructing and executing the simulation model, and for recording event data. |
| **Process** | Used to define a single process within the simulation model. Contains parameters for all process characteristics. |
| **Event** | Used to describe the characteristics of each event in the simulation. |
| **Buffer** | Used to take events from a process and store them in a buffer. |
| **DataCrawler** | Data analysis class that processes all collected event data to provide the initial analysis. |

#### *3.3.3.1 Class Structure*

Each of the five classes of the simulation engine of the ASAE text-based simulation framework provide specific functionality when running the simulation model of a MAS. Figure 3.13 depicts a Unified Modeling Language (UML) class diagram of the simulation engine. The UML class diagram shows the relationships between the

classes, as well as a more detailed description of the data elements and functions included in each class.

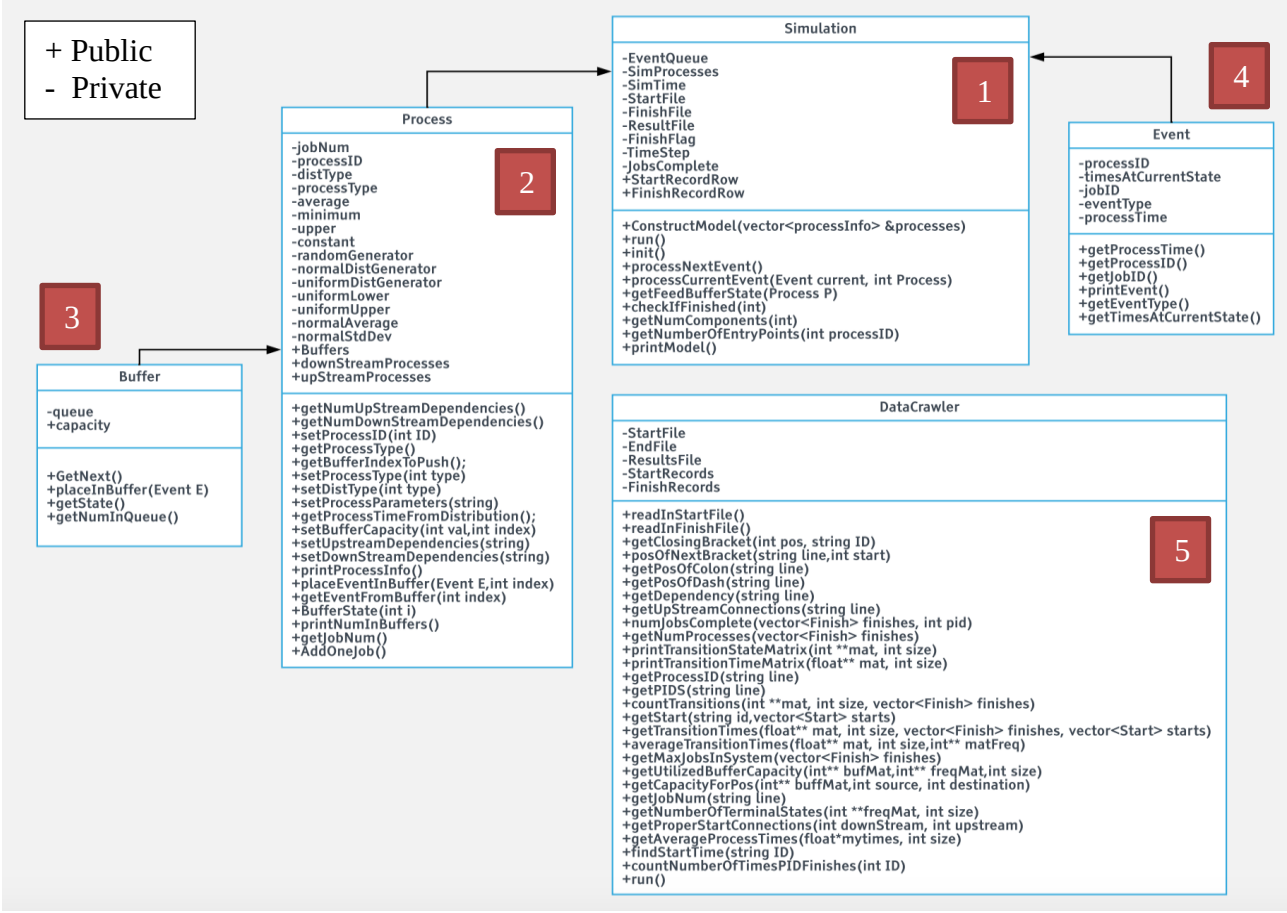



**Figure 3.13. UML Class Diagram of the Simulation Engine.**

Each class in the UML class diagram is represented as a block. Each block is divided into three sections, as depicted in Figure 3.14. The top section displays the name of the class. The second section is used to list the variables that are members of the class. Finally, the third section lists the functions of the class. To the left of each variable and function listed within a class block, a “+” or a “-” is used to indicate whether the variable or function is public or private to the class.

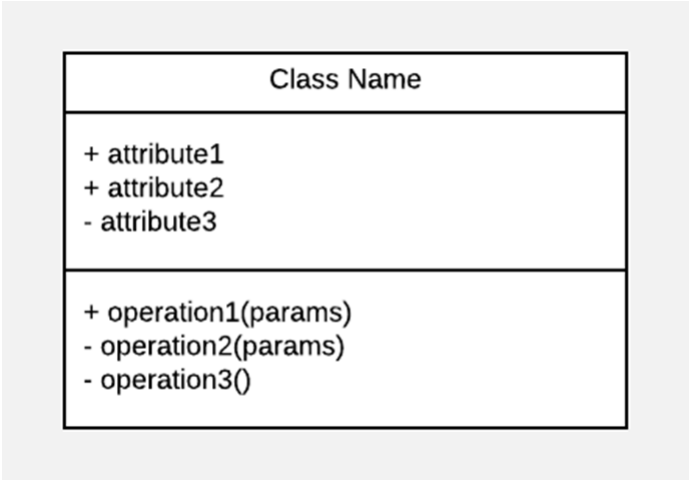



**Figure 3.14. Format of a UML Class Block.**

The following sections provide more details about the specific functionality provided by each of the five classes that comprise the simulation engine of the ASAE text-based simulation framework.

#### *3.3.3.2 Simulation Class*

Within the ASAE text-based simulation framework, the Simulation class is the main driver of the simulation as it contains all the time and scheduling information used to process the correct events in the proper order.

The Simulation class implements a priority queue data structure in which events are ordered based on their priority (i.e., the time the event is scheduled to take place). Events are continually processed and pulled from the priority queue in chronological order. The simulation time is controlled by the processing and scheduling of events in the priority queue. In the case of wait events (i.e., a process is blocked or starved), the simulation waits for a predefined time step of .001 time units until an entity is ready to be processed or space is available in a buffer. The Simulation class also contains the definition of each process within the MAS.

#### *3.3.3.3 Process Class*

The Process class is used to capture the unique characteristics of each process within the MAS, including process time, position type, upstream dependencies, downstream dependencies, and associated buffers.

As explained in Section 3.3.2, the ASAE text-based simulation framework is capable of simulating process times using the normal distribution, the uniform distribution, the triangular distribution, or a constant time. The position type parameter indicates whether a process is at the front, middle, or end of the line. The parameters for the upstream and downstream dependencies tell the system what

jobs come before and after a specific process. Each process is also always associated with buffers that will collect jobs when finished with processing.

#### *3.3.3.4 Buffer Class*

The Buffer class is used to represent the characteristics of a buffer between process steps. The characteristics of a buffer include a queue with a finite capacity to allow for jobs (i.e., events within the ASAE text-based simulation framework) to be stored. A buffer in the ASAE text-based simulation framework can be thought of as a first-in, first-out (FIFO) container with a certain amount of space.

In the context of C++, there is a data structure called a queue that represents a FIFO construct to store data objects. As a process finishes a job, this job is then placed in the queue. A downstream process can then pull that job from the buffer when it is ready to continue processing. The capacity of the buffer is also set in the Buffer class to describe how many jobs can be stored in the buffer.

In the ASAE text-based simulation framework, a buffer reports its own state to the rest of the simulation system. When called, a buffer can indicate that it is full, empty, or has space. The state of a buffer controls how the rest of the simulation will schedule events.

#### *3.3.3.5 Event Class*

Events are what drive the entire simulation. The Event class is used to represent a specific event that takes place within a process at a distinct time. There are six different types of events that can occur in a simulation, as shown in Table 3-8.

**Table 3-8. Types of Events in the ASAE Text-Based Simulation Framework.**

| Event Type | Description |
|---|---|
| **Start** | Used to represent the beginning of processing. |
| **Finish** | Used to represent the completion of processing. |
| **Push** | Used to represent the placement of a job into an output buffer. |
| **Pull** | Used to represent the extraction of a job from an input buffer. |
| **Wait to Push** | Used to indicate that the simulation must wait before placing another job in the output buffer. |
| **Wait to Pull** | Used to indicate that the simulation must wait before extracting another job from the input buffer. |

Each type of event listed in Table 3-8 has a specific consequence (i.e., schedules another event and places it into the event queue) based on the event type and the state of dependencies. The event processing logic followed in the ASAE text-based simulation framework is depicted in Figure 3.15.

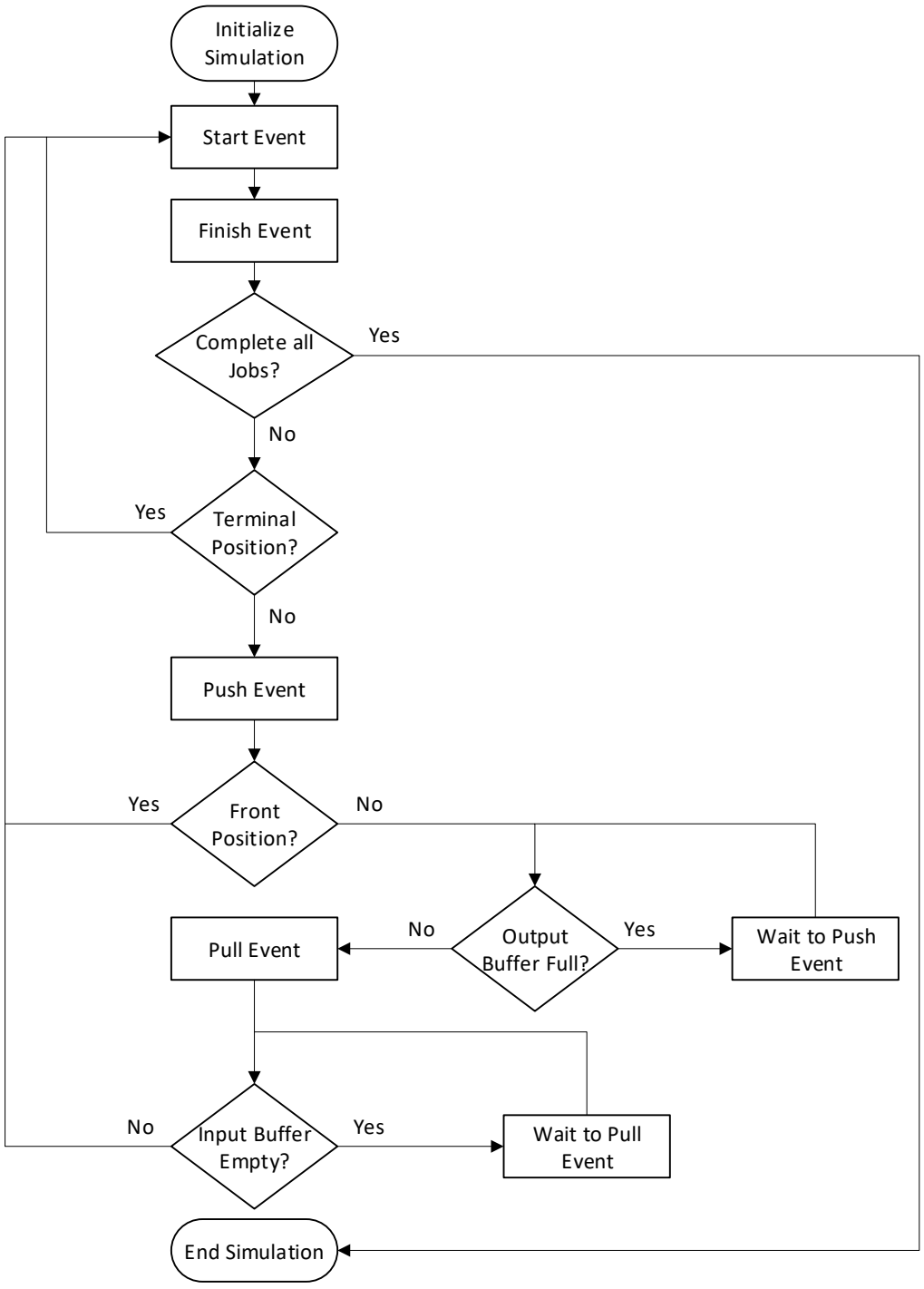



**Figure 3.15. Control Logic for Processing and Scheduling Events.**

In the ASAE text-based simulation framework, a “Start” event is scheduled to indicate that a process has begun. When a “Start” event is processed, a processing time is generated from a statistical distribution and added to the current simulation time to provide the scheduling time for a corresponding “Finish” event. The “Finish” event represents an entirely new event used to indicate the end of processing at a given process.

When processing a “Finish” event, a corresponding “Push” event is scheduled at the exact time of the “Finish” event. A “Push” event represents the placement of a job into a buffer. A “Push” event can schedule either a “Pull” event or a “Wait to Push” event based on the state of the process’s buffer. If the buffer that the event is supposed to be placed in is not full, then the event can be placed in the queue and a “Pull” event is scheduled. If the buffer is full, then a “Wait to Push” event is scheduled, and a standard simulation time increment of .001 time units is added to the time of the “Push” event. The “Wait to Push” event will then enter a cycle of incrementing the scheduled time by the simulation’s time interval until the buffer has space available. When the buffer has space, the “Wait to Push” event will then schedule a “Pull” event for the process to continue processing.

The “Pull” event represents the process of pulling the next job from the input buffer. When a “Pull” event is processed, the input buffer will be checked to indicate if a job is available. If a job is available, then it can be pulled from the queue and a “Start” event is scheduled. If a job is not available, then the process must wait to pull a job with a “Wait to Pull” event. The “Wait to Pull” event is

similar to the “Wait to Push” event in that the “Wait to Pull” event also enters into a cycle of incrementing the scheduled time until the input buffer has a job that can be pulled into the starved process.

When an event is at the front of the line and when an event is at the end of the line the standard processing model changes slightly. If the event is at the front of the line (i.e., there is no input buffer), a “Start” event is immediately scheduled from a “Push” event. If the event is at the end of the line (i.e., there is no output buffer) the “Pull” event is immediately scheduled from a “Finish” event.

#### *3.3.3.6 DataCrawler Class*

The DataCrawler class encompasses all the algorithmic analysis described in Section 3.2.3 within a single C++ class. The attributes in this C++ class are extracted from the “starts.txt” and “finish.txt” files, which are included in Appendix J and Appendix K respectively. The “starts.txt” and “finish.txt” text files are read into the program and converted into vectors (i.e., a C++ array data structure that contains additional functionality to easily manage items) containing the details of each start object and finish object. The start and finish objects are C++ structures that store data for easy access and processing. The DataCrawler C++ class then follows the sequence of data analysis steps described in Section 3.2.3. When each data analysis step is completed, the results are recorded in a results file and presented to the user.

#### *3.3.3.7 Running a Simulation with the ASAE Text-Based Framework*

Figure 3.16 depicts the steps required to execute a simulation using the ASAE text-based simulation framework. The first step is to read in and parse the model file provided by the user. More specifically, the model file is parsed to identify (1) the characteristics of each process within the simulation, and (2) the number of jobs to simulate, as specified by the user. The model description that results from parsing the model file is then used in the second step to construct the simulation model. The construction of the simulation model involves defining each process instance and connecting the processes together based on the model file provided by the user.

Once the simulation model is constructed, the next step is to initialize it with "Start" events (i.e., jobs). A "Start" event is scheduled for every position at the front of the line (i.e., an entry point into the process), which begins the event processing loop seen in Figure 3.15. Once the simulation is initialized, the simulation model can be executed, and events are pulled from the event queue. During event processing, "Start" events and "Finish" events are collected for use in analysis. Once the simulation has completed the required number of jobs, the simulation terminates.

At the completion of the simulation, start information and finish information for all complete jobs is stored in two files named "starts.txt" and "finish.txt", which are then used to initialize the DataCrawler class. Each of the recorded start records and finish records are read into the DataCrawler class for use

in processing. The DataCrawler class then uses these data to determine performance characteristics of the MAS being simulated. Each performance characteristic calculated by the DataCrawler class is written into a report, which is presented to the user before the program closes.

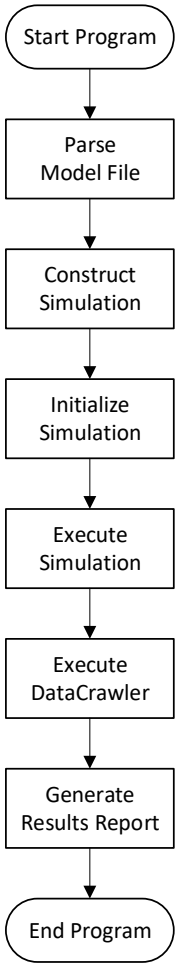



**Figure 3.16. Steps for running a Simulation in the ASAE Text-Based Simulation Framework.**

#### *3.3.3.8 Capturing Performance Data from the Simulation*

The DataCrawler class enables the collection of simulation performance data from "Start" and "Finish" events in a specific format. "Start" and "Finish" events are tracked during the simulation because they are fundamental in determining when jobs enter and exit either a single process steps or a series of linked process steps. Once the start and end of a process (or a group of processes) has been identified, the transfer times and buffer characteristics associated with these resources can be determined.

Each data point recorded from the simulation of a MAS must contain specific identifiers to allow the DataCrawler class to understand with which process the data point is associated. These identifiers include the job ID, the simulation time associated with the job ID, the dependencies associated with this job, and the number of jobs in the system. The dependency information must be included to understand process flow. For example, if a process has two upstream dependencies, then the collected data point must reflect that both of those dependencies were included in that particular process. It should be noted that a job for a particular process is represented by the combination of one "Pull" event, one "Start" event, one "Finish" event, one "Push" event, and, potentially, many "Wait to Pull/Push" events. The set of events will contain the same job instance number for a given job at a process.

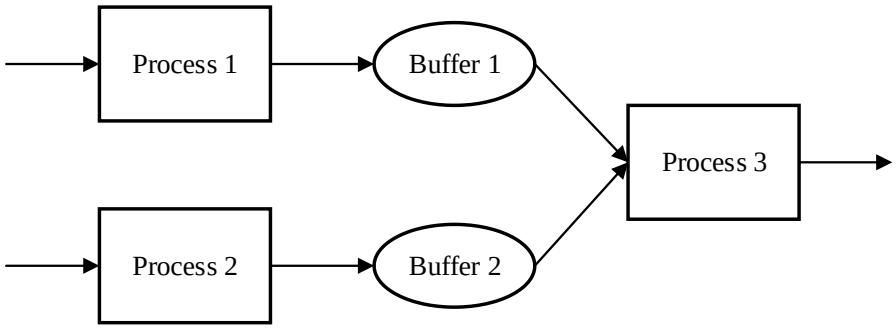
The ASAE text-based simulation framework records event data in the "start.txt" and "finish.txt" data files using the following format:

**[Job ID], Current Simulation Time, Number of Jobs in System**

The format of the field [Job ID] is crucial to track process flow and must comply with the following format:

**[Job_Instance:Process_ID-(Dependencies)]**

Figure 3.17 depicts an example of how the strings shown above would be formatted for a MAS Type II. A complete example of the "start.txt" and "finish.txt" text files are included in Appendix J and Appendix K respectively. It is important to note that if a specific job does not contain dependencies, then the (Dependencies) field is populated with an "x", as illustrated in Figure 3.17 for the strings of Process 1 and Process 2.




Data Point for Job 1, Process 1 → [1:1-(x)],0.000000,1
Data Point for Job 1, Process 2 → [1:2-(x)],0.000000,2
Data Point for Job 1, Process 3 → [1:3-([1:1-(x)][1:2-(x)])],5.644773,4

**Figure 3.17. Example Data Entries for One Complete Job.**

*3.3.3.9 Analysis Report*

The ASAE text-based simulation framework produces a final report that is presented to the user once the simulation of a MAS is completed. An example of this report is depicted in Figure 3.18.

The top of the final report file lists the simulation time, the number of start and finish records that were used in the data analysis, and the total simulation runtime. It is important to note that all the time-based metrics included in the final report are expressed in general time units. The final report also includes information about the throughput for each terminal state and the total overall throughput of the MAS, as well as the maximum (max) number of components in the system (i.e., work-in-process).

The lower portion of the results file focuses on the TSM and its representation in terms of frequencies, transition times, and maximum utilized buffer capacity. The report file finishes with an overview of each processing time.

```
Results for Tue Oct  2 21:29:46 2018

Number of Start Records: 308
Number of Finish Records: 307

Simulation runtime: 608.242

There are/is 1 Terminal State(s)
-Number of Jobs Exiting via PID 2: 100
-Average time to produce one job: 6.08242

Total Complete Jobs: 100
Overall average time per job: 6.08242

Max Number of Components in System: 9
Number of Processes: 3

Transition State Matrix
PID   0   1   2
  0|  0|  0|100|
  1|  0|  0|100|
  2|  0|  0|  0|

Total Transition Times Matrix
      PID          0          1          2
        0|    0.000|    0.000| 2993.744|
        1|    0.000|    0.000|    0.049|
        2|    0.000|    0.000|    0.000|

Average Transition Times Matrix
      PID          0          1          2
        0|    0.000|    0.000|   29.937||
        1|    0.000|    0.000|    0.000|
        2|    0.000|    0.000|    0.000|

Max Utilized Buffer Capacity Matrix
PID   0   1   2
  0|  0|  0|  5|
  1|  0|  0|  0|
  2|  0|  0|  0|

Average Process Times
       0        1        2
   3.998|   6.034|   3.721|
```

**Figure 3.18. Example Results File.**

## 3.4 TESTING AND VALIDATION OF ASAE TEXT-BASED SIMULATION FRAMEWORK

Several tests were conducted to ensure that the design and implementation of the ASAE text-based simulation framework were done properly and correctly. A set of tests focused on assessing the correctness of the ASAE text-based simulation framework. In addition, a user study was conducted to collect unbiased data about the usefulness and practicality of the ASAE text-based simulation framework.

### 3.4.1 Study to Validate Correctness

The ASAE text-based simulation framework uses a complex logic system that enforces how random events (i.e., based on statistical simulations) are processed and scheduled in a simulation model. To validate the correctness of the ASAE text-based simulation framework (i.e., its ability to accurately process events and simulate a MAS), all three types of MASs were modeled and simulated using both Arena and the ASAE text-based simulation framework.

To keep the simulation run times reasonable in the study to validate the correctness of the ASAE text-based simulation framework, a different number of jobs were simulated for each type of MAS as shown by Table 3-9.

**Table 3-9. Number of Jobs Simulated per MAS Type.**

| MAS Type | Jobs Simulated |
|---|---|
| I | 100 |
| II | 50 |
| III | 20 |

The simulation model of each type of MAS was replicated 100 times in both Arena and the ASAE text-based simulation framework. A slightly modified version of the ASAE text-based simulation framework was created to include a log system that would record the simulation runtime in a CSV file when the simulation finished. This version of the ASAE text-based simulation framework was then integrated into a test script written in the Python programming language that executed the simulation of each MAS type 100 times, thus producing 100 simulation runtimes. In Arena, the simulation model of each MAS type was set to run for 100 replications and VBA code was used to write the simulation time into an MS Excel spreadsheet after each replication was complete.

The runtimes obtained from each of the three types of MASs using Arena and the ASAE text-based simulation framework were then used to produce scatter plots and box-and-whisker plots. Furthermore, F-tests were conducted to test the hypothesis that variances of the runtimes collected with the ASAE text-based simulation framework and Arena were equal.

A second test to validate the correctness of the ASAE text-based simulation framework involved mapping a MAS Type II manually using 20 jobs with constant times. In this test, the behavior of the MAS Type II was analyzed in one-minute increments to extract the number of jobs in buffers and the number of start and finish events. The finish time of the simulation was also recorded. The results obtained from manually mapping the behavior of the MAS Type II were then compared against the results obtained from simulating 20 jobs with constant times through the same type of MAS using both the ASAE text-based simulation framework and Arena. Finally, 100 jobs with constant times were then simulated within the ASAE text-based simulation framework and Arena to provide further support in validating the control logic of the ASAE text-based simulation framework.

### 3.4.2 User Study

While the correctness of the ASAE text-based simulation framework is important, it is also critical to understand the usability and the potential value that the ASAE text-based simulation framework brings to the end user. As stated before, the main objective of the ASAE text-based simulation framework is to simplify and streamline the simulation process, thus making it easier for a user with any level of experience to quickly create simulation models and understand the characteristics of a MAS. While it is hypothesized that the ASAE text-based simulation framework meets this objective, this claim had to be tested and confirmed.

#### *3.4.2.1 Objective*

While usability was tested throughout the development of the ASAE text-based simulation framework, the testing was not done with unbiased users. Therefore, the objective of the user study was to collect feedback from unbiased users about the usability and potential value of the ASAE text-based simulation framework. The user study was helpful in identifying how the ASAE text-based simulation framework may improve the simulation workflow, and what opportunities exist to improve the performance of this new tool.

In the user study, unbiased subjects were exposed to a simulation exercise that utilized the well-known, GUI-based Arena simulation software as well as the ASAE text-based simulation framework. Upon completing the simulation exercise, a questionnaire was administered to the study participants to collect their feedback on different aspects of the simulation process. The user study conducted in this research was reviewed and approved by the Institutional Review Board (IRB). The approval notice received by the IRB is included in Appendix G.

#### *3.4.2.2 Questionnaire Design*

The questionnaire that helped to assess the usability and potential value of the ASAE text-based simulation framework was organized into nine blocks. These blocks are detailed in Table 3-10. To prevent bias toward the ASAE text-based simulation framework or toward Arena, each question in a block has an equivalent

counter question. Qualtrics was used as the platform to implement and administer the user study questionnaire.

**Table 3-10. Question Blocks of the User Study Questionnaire.**

| Block Focus | Description |
|---|---|
| **Experience** | The questions in this block tried to assess the level of experience a user had using simulation and related technologies. |
| **Modeling** | The questions in this block focused on the process of modeling a MAS with the ASAE text-based simulation framework and the Arena DES software and how the two different approaches compared in terms of time and usability. |
| **Running a Simulation** | The questions in this block tried to assess the impact each software system has on the time it takes to run a simulation model using the ASAE text-based simulation framework and the Arena simulation software. |
| **GUI-based vs Text-based** | The questions in this block tried to assess a user's experience when modeling with a GUI-based system vs a text-based system to understand the strengths and weaknesses of each approach. |
| **Parallel Processes and Finite Buffers** | The questions in this block tried to assess the ability of the ASAE text-based simulation framework and the Arena DES software to effectively model parallel workstations and finite capacity buffers. |
| **Understanding the MAS** | The questions in this block are used to assess how well the ASAE text-based simulation framework and the Arena DES software allow a user to understand characteristics of a MAS. |
| **Value of Results** | The questions in this block tried to assess the value of the results provided by the ASAE text-based simulation framework and the Arena DES software. More specifically, does each software system allow a user to understand certain critical performance characteristics of a MAS. |
| **Compare Simulation Platforms** | The questions in this block tried to assess the elements of a simulation exercise in terms of how well certain points align with either ASAE or Arena. |
| **Preference** | The questions in this block tried to assess generally how many users prefer each software system. |

#### *3.4.2.3 Recruitment of Participants*

The target population to recruit participants for the user study included both undergraduate and graduate students at Oregon State University (OSU), regardless of age, with at least some exposure to simulation technologies. In particular, students with prior experience with Arena were encouraged to participate.

Different methods were used to recruit participants for the user study, including an email message posted to listservs (see Appendix E) and flyers posted at different locations at OSU (see Appendix F). In the end, 31 students participated and completed the user study.

#### *3.4.2.4 User Participation and Interaction*

The user study participants were involved in the four-step study protocol described in Appendix H.

As a first step, the user study participants were introduced to the ASAE text-based simulation framework and to Arena to gain a basic understanding of the components of each system and to learn how to create simulation models successfully. In step 2, the study participants completed a short exercise using the ASAE text-based simulation framework which involved creating a model of a MAS Type II and simulating 100 jobs. In step 3, the study participants created a model of the same MAS Type II, but with Arena and also simulated 100 jobs.

The last step of the user study protocol involved administering the questionnaire described in Section 3.4.2.2 to the participants where they were asked

to provide input on their experience using the ASAE text-based simulation framework and Arena.

# 4 RESULTS AND DISCUSSION

This chapter presents and discusses the results of the tests conducted in this research to validate that the design and implementation of the ASAE text-based simulation framework were done properly and correctly.

Section 4.1 presents and discusses the results of the quantitative validation of the correctness of the ASAE text-based simulation framework based on data collected from simulations of all three types of MASs using both Arena and the ASAE text-based simulation framework.

Section 4.2 then presents and discusses the results of the qualitative assessment conducted through a user study, which allowed the collection of unbiased data about the usefulness and practicality of the ASAE text-based simulation framework.

## 4.1 RESULTS OF VALIDATING THE CORRECTNESS OF THE ASAE TEXT-BASED SIMULATION FRAMEWORK

The proper execution of a DES simulation is largely dependent on the accuracy with which random numbers from known statistical distributions are generated. Therefore, testing was completed to measure the degree of randomness of the simulation engine of the ASAE text-based simulation framework.

### 4.1.1 Simulation Runtimes Results

The simulation runtime for each of the 100 replications of each type of MAS collected with both Arena and the ASAE text-based simulation framework was recorded in a CSV file. Scatter plots and box-and-whisker plots of these simulation runtimes were created in MS Excel to assess their randomness. In addition, statistics were calculated across the 100 replications for each type of MAS and used in an F-test to test the hypothesis that the variances of the runtimes produced by the ASAE text-based simulation framework and Arena were equal. The complete set of raw data for each test is included in Appendix L. The results obtained for each MAS type are presented and discussed in the next sections.

#### *4.1.1.1 MAS Type I*

Figure 4.1 depicts the scatter plot of the simulation runtime of each of the 100 replications for a MAS Type I obtained with the ASAE text-based simulation framework and Arena. The results obtained with the ASAE text-based simulation framework are represented with blue circles, whereas the results obtained with Arena are represented with red triangles. As shown in Table 3-9, 100 jobs were simulated for each replication of a MAS Type I.

The simulation runtimes generated by the simulation engine of the ASAE text-based simulation framework plotted in Figure 4.1 appear random and do not exhibit any concerning trends. Figure 4.2 shows a box-and-whisker plot of the data, which further shows the similarity of the two data sets.

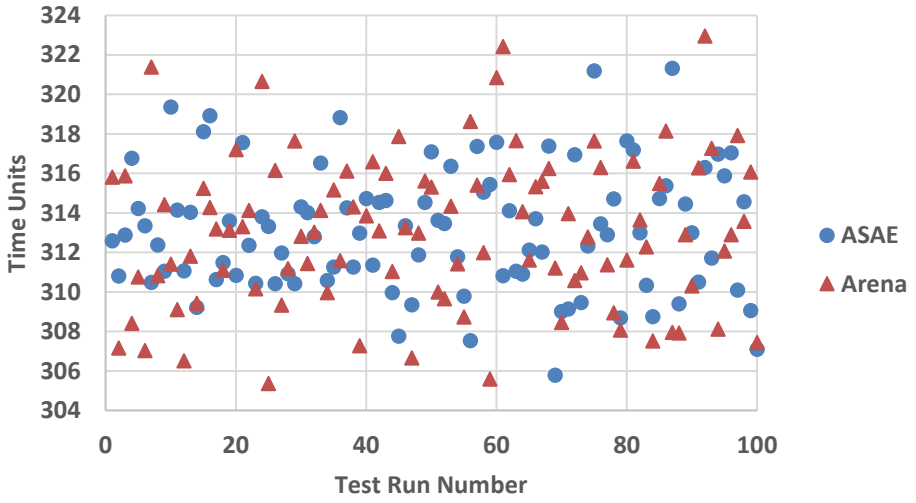



**Figure 4.1. Scatter Plot of Simulation Runtimes for the MAS Type I.**

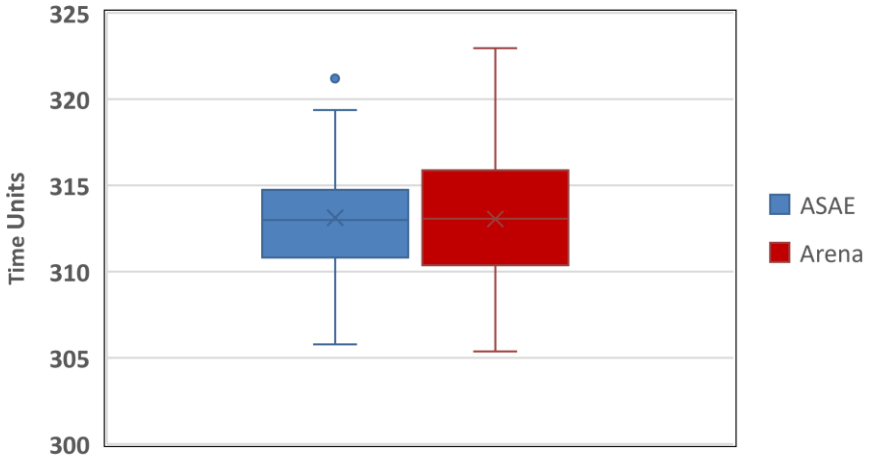



**Figure 4.2. Box and Whisker Plot of Simulation Runtimes for MAS Type I.**

The values for the average, standard deviation, and the variance calculated from the 100 simulation runtimes obtained for the MAS Type I with both Arena and the ASAE text-based simulation framework are shown in Table 4-1.

**Table 4-1. Statistics of the Simulation Runtimes for the MAS Type I.**

| | Average | Standard Deviation | Variance |
|---|---|---|---|
| **ASAE** | 313.11 | 3.14 | 9.84 |
| **Arena** | 313.05 | 3.81 | 14.55 |

An F-test was conducted in Minitab to test the hypothesis that the variances of the runtimes produced by the ASAE text-based simulation framework and Arena for the MAS Type I were equal ($H_o$: $\sigma^2_{ASAE} = \sigma^2_{Arena}$) versus not being equal ($H_a$: $\sigma^2_{ASAE} \neq \sigma^2_{Arena}$). The p-value obtained from the F-test was 0.053, which indicates that the null hypothesis (i.e., $\sigma^2_{ASAE} = \sigma^2_{Arena}$) cannot be rejected at a significance level of $\alpha = 0.05$.

#### *4.1.1.2 MAS Type II*

Figure 4.3 depicts the scatter plot of the simulation runtime of each of the 100 replications for a MAS Type II obtained with the ASAE text-based simulation framework and Arena. The results obtained with the ASAE text-based simulation framework are represented with blue circles, whereas the results obtained with Arena are represented with red triangles. As shown in Table 3-9, 50 jobs were simulated for each replication of a MAS Type II.

The simulation runtimes generated by the simulation engine of the ASAE text-based simulation framework plotted in Figure 4.3 appear random and do not

exhibit any concerning trends. Figure 4.4 shows a box-and-whisker plot of the data, which further shows the similarity of the two data sets.

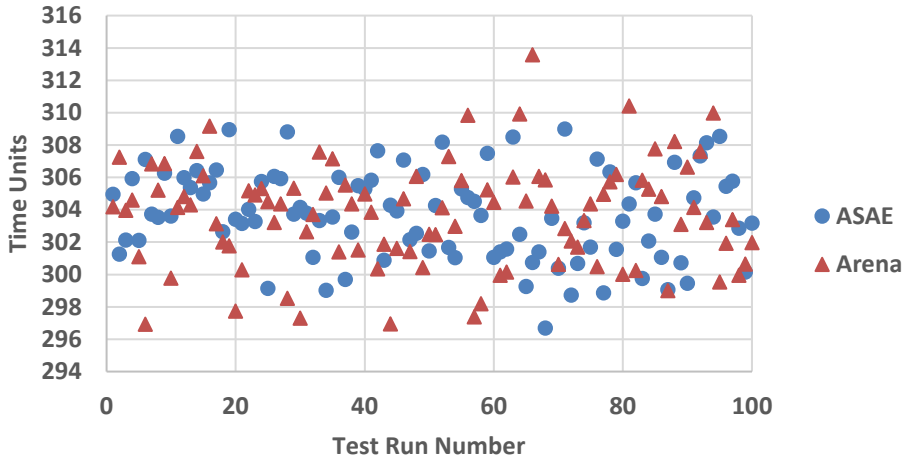



**Figure 4.3. Scatter Plot of Simulation Runtimes for the MAS Type II.**

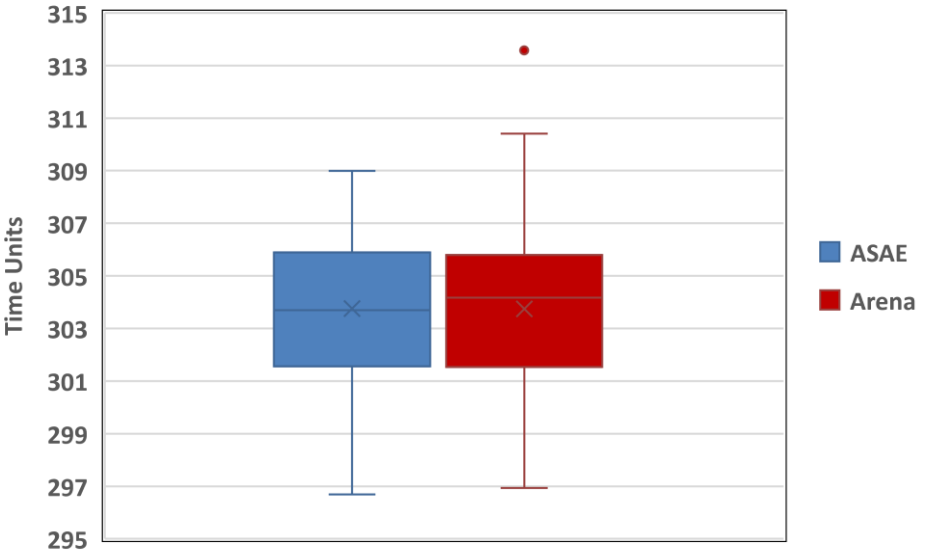



**Figure 4.4. Box and Whisker Plot of Simulation Runtimes for MAS Type II.**

The values for the average, standard deviation, and the variance calculated from the 100 simulation runtimes obtained for the MAS Type II with both Arena and the ASAE text-based simulation framework are shown in Table 4-2.

**Table 4-2. Statistics of the Simulation Runtimes for the MAS Type II.**

| | **Average** | **Standard Deviation** | **Variance** |
|---|---|---|---|
| **ASAE** | 303.76 | 2.76 | 7.63 |
| **Arena** | 303.74 | 3.21 | 10.31 |

An F-test was conducted in Minitab to test the hypothesis that the variances of the runtimes produced by the ASAE text-based simulation framework and Arena for the MAS Type II were equal ($H_o$: $\sigma^2_{ASAE} = \sigma^2_{Arena}$) versus not being equal ($H_a$: $\sigma^2_{ASAE} \neq \sigma^2_{Arena}$). The p-value obtained from the F-test was 0.136, which indicates that the null hypothesis (i.e., $\sigma^2_{ASAE} = \sigma^2_{Arena}$) cannot be rejected at a significance level of $\alpha = 0.05$.

#### *4.1.1.3 MAS Type III*

Figure 4.5 depicts the scatter plot of the simulation runtime of each of the 100 replications for a MAS Type III obtained with the ASAE text-based simulation framework and Arena. The results obtained with the ASAE text-based simulation framework are represented with blue circles, whereas the results obtained with

Arena are represented with red triangles. As shown in Table 3-9, 20 jobs were simulated for each replication of a MAS Type III.

The simulation runtimes generated by the simulation engine of the ASAE text-based simulation framework plotted in Figure 4.5 appear random and do not exhibit any concerning trends. Figure 4.6 shows a box-and-whisker plot of the data, which further shows the similarity of the two data sets.

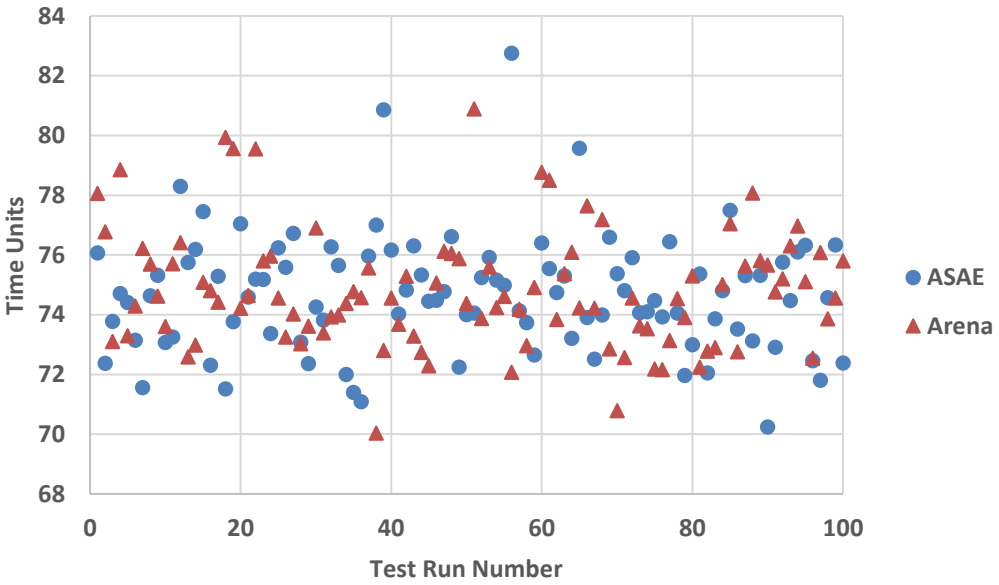



**Figure 4.5. Scatter Plot of Simulation Runtimes for the MAS Type III.**

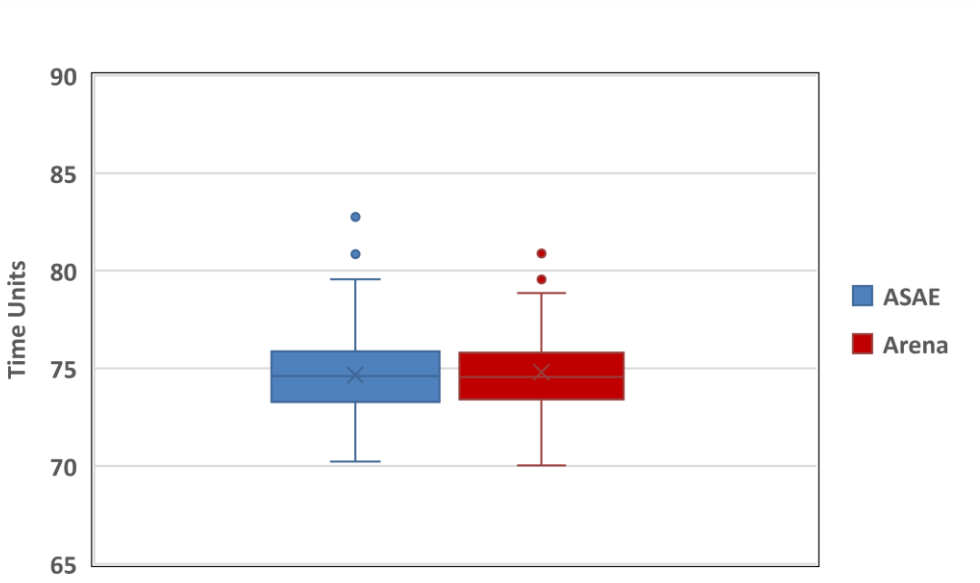



**Figure 4.6. Box and Whisker Plot of Simulation Runtimes for MAS Type III.**

The values for the average, standard deviation, and the variance calculated from the 100 simulation runtimes obtained for the MAS Type II with both Arena and the ASAE text-based simulation framework are shown in Table 4-3.

**Table 4-3. Statistics of the Simulation Runtimes for the MAS Type III.**

| | Average | Standard Deviation | Variance |
|---|---|---|---|
| **ASAE** | 74.76 | 1.97 | 3.90 |
| **Arena** | 74.81 | 1.97 | 3.88 |

An F-test was conducted in Minitab to test the hypothesis that the variances of the runtimes produced by the ASAE text-based simulation framework and Arena

for the MAS Type III were equal ($H_o$: $\sigma^2_{ASAE} = \sigma^2_{Arena}$) versus not being equal ($H_a$: $\sigma^2_{ASAE} \neq \sigma^2_{Arena}$). The p-value obtained from the F-test was 0.979, which indicates that the null hypothesis (i.e., $\sigma_{ASAE} = \sigma_{Arena}$) cannot be rejected at a significance level of $\alpha = 0.05$.

### 4.1.2 Manually Mapped MAS Results

The simulation engine of the ASAE text-based simulation framework utilizes a complex set of rules to dictate how events are scheduled and processed based on various dependencies and times. The objective of this phase of the testing was to validate the correctness of the control logic implemented by the ASAE simulation engine to ensure that events were being scheduled and processed correctly. In performing this testing, it was not practical to simulate a large MAS due to the process complexities involved. Therefore, the validation was performed by manually mapping 20 jobs with constant time for a MAS Type II consisting of three processes (i.e., Process 0, Process 1, and Process 2) and two finite buffers (i.e., Buffer 0 and Buffer 1). The results obtained from manually mapping the MAS Type II were then compared to the results obtained from simulating the exact same MAS Type II using the ASAE text-based simulation framework.

Figure 4.7 depicts the results of manually mapping the MAS Type II. Each job is tracked through the system using a specific color. The main advantage of manually mapping the execution of the MAS Type II is that every event and state

of the system is known before simulating the same MAS Type II using the ASAE text-based simulation framework to verify the behavior of the simulation.

By manually mapping the MAS, several characteristics of the simulation can be calculated precisely including the simulation runtime, maximum utilized buffer capacity, and the number of start/finish events. For example, the time scale at the top of Figure 4.7 shows that the total runtime for this specific MAS Type II was 62 time units. The maximum utilized buffer capacity can be calculated by observing how many jobs are in the buffers, as reflected in the Buffer 0 and Buffer 1 rows. In this example, each buffer has a capacity of three. Therefore, three slots have been labeled “B” (i.e., back), “M” (i.e., middle), and “F” (i.e., front) in Figure 4.7. The start events are calculated by counting the total number of job blocks within each process row that start within the running simulation time (i.e., any block that is visible within the available 62 time units available). Similarly, the finish events are calculated by counting the total number of job blocks that actually finish within the simulation time (i.e., 62 time units).

Table 4-4 summarizes the results of manually mapping the MAS Type II and also presents the results of simulating the same system with the ASAE text-based simulation framework. These results confirm that the ASAE simulation engine is processing and scheduling events correctly since the values of the simulation parameters are identical in both cases.

**Table 4-4. Predicted vs Actual Simulation Characteristics.**

| Simulation Parameters | Manual Mapping | ASAE Simulation |
|---|---|---|
| **Runtime (Time Units)** | 62 | 62 |
| **Max Utilized Capacity of Buffer 0 (Count)** | 0 | 0 |
| **Max Utilized Capacity of Buffer 1 (Count)** | 3 | 3 |
| **Number of Start Events (Count)** | 65 | 65 |
| **Number of Finish Events (Count)** | 64 | 64 |

Table 4-5 shows the results of simulating 20 jobs and 100 jobs with the ASAE text-based simulation framework and Arena. The time to complete the simulation of the 20 and 100 jobs in a MAS Type II were identical for both DES systems.

**Table 4-5. Simulation Runtime in ASAE and Arena.**

| | Time Units | |
|---|---|---|
| | **ASAE** | **Arena** |
| **Time to Complete 20 Jobs** | 62 | 62 |
| **Time to Complete 100 Jobs** | 302 | 302 |

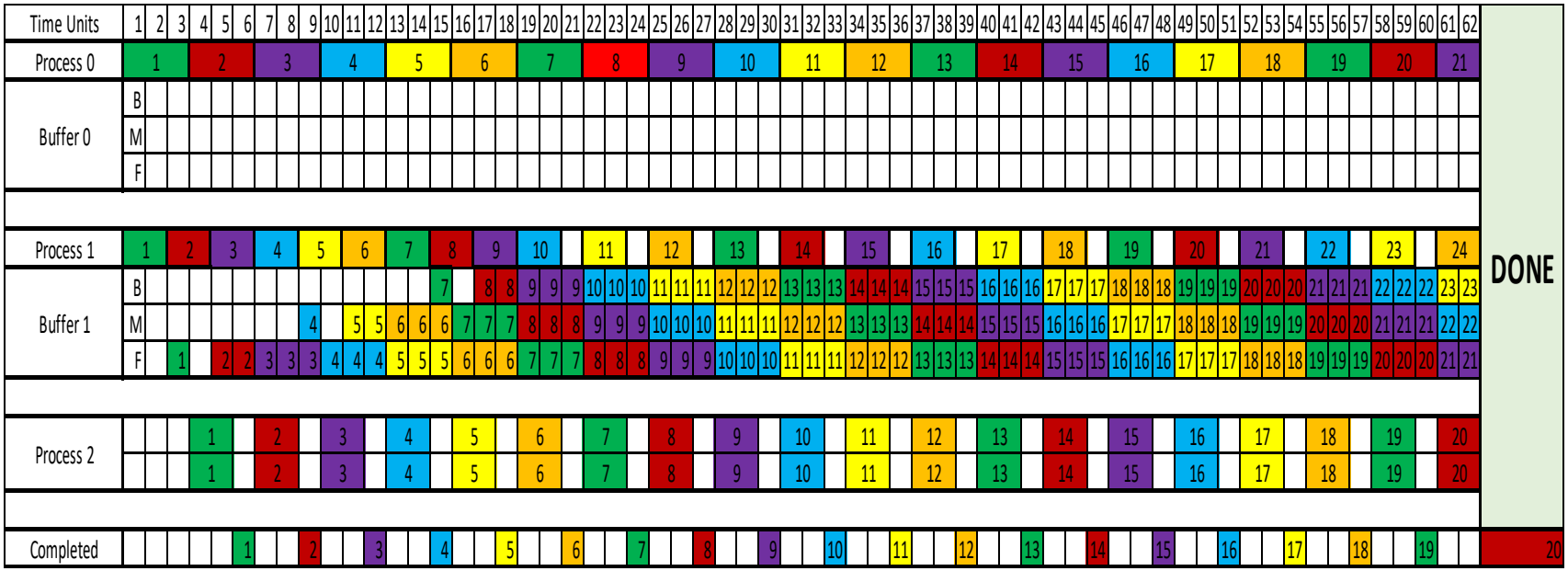



**Figure 4.7. Manually Mapped Simulation of 20 Jobs in a MAS Type II.**

## 4.2 QUESTIONNAIRE RESULTS FROM THE USER STUDY

This section presents and discusses the results of the questionnaire conducted as part of the user study. As explained in Section 3.4.2.2, the questionnaire was administered to the user study participants *after* they had completed the pre-study training exercises on Arena and the ASAE text-based simulation framework. The main objective of the questionnaire was to gather data from a general population of users (with at least some exposure to simulation technologies) about the usability and value of the ASAE text-based simulation framework.

Table 3-10 shows the nine blocks that composed the questionnaire. Except from block one and block nine, every other block in the questionnaire was composed of a set of statements that the user study participants were asked to rate using a six-point Likert scale (i.e., strongly agree, agree, somewhat agree, somewhat disagree, disagree, and strongly disagree).

The following sections present and discuss the results of each block of the questionnaire.

### 4.2.1 Experience

A total of 31 participants responded to the questionnaire administered as part of the user study. In the first block of the questionnaire, the user study participants were asked to rate their ability to develop simulation models after they had completed the pre-study training exercises on Arena and the ASAE text-based simulation

framework. The responses received for this question block are shown in Table 4-6, and indicate that the majority of the participants (22 out of 31) rated their ability to develop simulation models as either “Good” (13 out of 31) or “Average” (9 out of 31). The remaining nine respondents rated their ability level as either “Excellent” (7 out of 31) or “Poor” (2 out of 31).

**Table 4-6. Overall Simulation Proficiency of User Study Participants.**

| | **Responses** | |
|---|---|---|
| **Proficiency Level** | **Count** | % |
| Excellent | 7 | 22.58% |
| Good | 13 | 41.94% |
| Average | 9 | 29.03% |
| Poor | 2 | 6.45% |
| **Total** | **31** | **100.00%** |

The counts per proficiency level shown in Table 4-6 demonstrate that the user study participants recruited in this research stated a diverse range of abilities in developing simulation models *after* they had completed the pre-study training exercises on Arena and the ASAE text-based simulation framework. Of particular interest are the nearly 71% (i.e., 22 out of 31) of user study participants which rated their ability to develop simulation models as either “Good” or “Average” because their feedback was considered very valuable when assessing how well the ASAE

text-based simulation framework is able to accomplish the main objective of this research, i.e., simplifying the complexity of simulation software solutions to allow for workers with an average understanding of simulation technologies to effectively employ these techniques.

### 4.2.2 Modeling

As Table 4-7 shows, the second block of the questionnaire was comprised of three statements that focused on assessing the impact that the ASAE text-based simulation framework and Arena have on the time needed by the user study participants to construct a model of a MAS.

**Table 4-7. Block 2 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Constructing a model of a manufacturing assembly system took less time with ASAE than with Arena. |
| 2 | ASAE reduced the time needed to construct a model of a manufacturing assembly system when compared to Arena. |
| 3 | ASAE has no impact on the time needed to construct a model of a manufacturing assembly system. |

Table 4-8 shows the responses received for the statements in this block of the user study questionnaire. The responses received for statements 1 and 2 were consistent in that 100% of the user study participants agreed in both cases with the premise that the ASAE text-based simulation framework saves time when

constructing the model of a MAS when compared to Arena. As stated before, the ability to save time and reduce the project timeline was one primary objective of the ASAE text-based simulation framework. The responses to statement 3 also suggest that the majority of user study participants (i.e., 27 out of 31, or 87.10%) agreed in that the ASAE text-based simulation framework has an impact on the time needed to construct a model of a MAS.

**Table 4-8. Overall Responses of User Study Participants on Modeling.**

| | Statement 1 Responses | | Statement 2 Responses | | Statement 3 Responses | |
|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 23 | 74.19% | 22 | 70.97% | 0 | 0.00% |
| Agree | 7 | 22.58% | 8 | 25.81% | 4 | 12.90% |
| Somewhat agree | 1 | 3.23% | 1 | 3.23% | 0 | 0.00% |
| Somewhat disagree | 0 | 0.00% | 0 | 0.00% | 3 | 9.68% |
| Disagree | 0 | 0.00% | 0 | 0.00% | 13 | 41.94% |
| Strongly disagree | 0 | 0.00% | 0 | 0.00% | 11 | 35.48% |

### 4.2.3 Running a Simulation

As shown in Table 4-9, the third block of the questionnaire was comprised of three statements that focused on assessing the impact that the ASAE text-based simulation framework and Arena have on the time it takes to simulate the model of a MAS.

**Table 4-9. Block 3 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Simulating a manufacturing assembly system took less time with Arena than with ASAE. |
| 2 | When compared to Arena, ASAE reduced the time needed to simulate a manufacturing assembly system. |
| 3 | When compared to Arena, ASAE has no impact on the time needed to simulate a manufacturing assembly system. |

Table 4-10 shows the responses received for the statements in this block of the user study questionnaire. The responses received for statement 1 clearly show that the majority of the user study participants (i.e., 27 out of 31, or 87.10%) disagreed with the statement that simulating the model of a MAS took less time with Arena than with the ASAE text-based simulation framework. Similarly, the responses received for statement 2 show that the majority of the user study participants (i.e., 28 out of 31, or 90.32%) agreed with the statement that the ASAE text-based simulation framework reduced the time needed to simulate the model of

a MAS. Finally, the responses for question 3 show that 87.10% (i.e., 27 out of 31) of the user study participants disagreed at some level (i.e., Strongly disagree", "Disagree", or "Somewhat disagree") in that ASAE text-based simulation framework (when compared to Arena) has no impact on the time needed to simulate a MAS. These results collectively suggest that the ASAE text-based simulation framework may in fact reduce the time needed to run the simulation of a MAS.

**Table 4-10. Overall Responses of User Study Participants on Running a Simulation.**

| | Statement 1 Responses | | Statement 2 Responses | | Statement 3 Responses | |
|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 3 | 9.68% | 19 | 61.29% | 1 | 3.23% |
| Agree | 1 | 3.23% | 6 | 19.35% | 1 | 3.23% |
| Somewhat agree | 0 | 0.00% | 3 | 9.68% | 2 | 6.45% |
| Somewhat disagree | 3 | 9.68% | 1 | 3.23% | 6 | 19.35% |
| Disagree | 12 | 38.71% | 2 | 6.45% | 11 | 35.48% |
| Strongly disagree | 12 | 38.71% | 0 | 0.00% | 10 | 32.26% |

### 4.2.4 GUI-based versus Text-based Modeling

As shown in Table 4-11, the fourth block of the questionnaire was comprised of six statements that focused on assessing the ease or difficulty experienced by the user study participants when constructing and simulating a MAS using a GUI-based system (i.e., Arena) versus a text-based system (i.e., ASAE text-based simulation framework). Statements 1, 2 and 3 focused on the ease of use of the ASAE text-based simulation framework and Arena, whereas statements 4 and 5 focused on capturing the preference of the participant between the two simulation software options used to construct and simulate a MAS.

**Table 4-11. Block 4 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Using Arena's GUI to construct and simulate a manufacturing assembly system is easier than using ASAE's text-based interface. |
| 2 | Using ASAE's text-based interface to construct and simulate a manufacturing assembly system is easier than using Arena's GUI. |
| 3 | Using ASAE's text-based interface to construct and simulate a manufacturing assembly system is more difficult than using Arena's GUI. |
| 4 | I would rather construct and simulate a manufacturing assembly system using Arena's GUI instead of ASAE's text-based interface. |
| 5 | I would rather construct and simulate a manufacturing assembly system using ASAE's text-based interface instead of Arena's GUI. |
| 6 | Both Arena's GUI and ASAE's text-based interface are equivalent when constructing and simulating a manufacturing assembly system. |

Table 4-12 shows the responses received for the statements in this block of the user study questionnaire. Figure 4.8 depicts the same results in a graphical format to aid in their interpretation.

The responses received for statement 1 and statement 2 appear to be contradictory. While the user participants were divided about how they rated their experience in using Arena's GUI-based approach to construct and simulate a MAS (i.e., statement 1), they rated their experience more favorably when completing the same tasks using the ASAE text-based simulation framework (i.e., statement 2), as evidenced by the 64.52% (i.e., 20 out of 31) of the user participants who chose either "Strongly agree", "Agree", or "Somewhat agree". The responses received for statement 3 seem to also validate those observed for statement 2, since 22 out of 31 participants (i.e., 70.96%) chose either "Strongly disagree", "Disagree", or "Somewhat disagree" when asked whether using the ASAE text-based interface to construct and simulate a MAS was more difficult than using Arena's GUI-based approach. In conclusion, the responses received for statement 1, statement 2, and statement 3 suggest that the user participants more often thought the ASAE text-based simulation framework was easier to use than the GUI provided by Arena.

The responses received for statement 4 and statement 5 proved to be difficult to interpret. In both cases, the opinions of the user study participants about whether they would prefer to use Arena's GUI or ASAE text-based simulation

framework when constructing and simulating a MAS were inconclusive. When responding to statement 4, 43.33% of the user study participants agreed on some level and 56.67% disagreed on some level. Similarly, 58.06% of the user study participants agreed on some level and 41.94% disagreed on some level when responding to statement 5. It is important to note that one participant did not provide a response for statement 4, bringing the total number of responses for this statement to 30.

Finally, statement 6 was an unbiased statement to assess if the user study participants believed the two modeling approaches to be equivalent. In this case, the user study participants were very clear in that the majority disagreed on some level (i.e., 74.19%) with the premise that Arena's GUI and ASAE's text-based interface are equivalent when constructing and simulating a MAS.

**Table 4-12. Overall Responses of User Study Participants on GUI-based vs Text-based Modeling.**

| | Statement 1 Responses | | Statement 2 Responses | | Statement 3 Responses | | Statement 4 Responses | | Statement 5 Responses | | Statement 6 Responses | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 1 | 3.23% | 5 | 16.13% | 1 | 3.23% | 3 | 10.00% | 5 | 16.13% | 2 | 6.45% |
| Agree | 4 | 12.90% | 7 | 22.58% | 4 | 12.90% | 1 | 3.33% | 7 | 22.58% | 3 | 9.68% |
| Somewhat agree | 8 | 25.81% | 8 | 25.81% | 4 | 12.90% | 9 | 30.0% | 6 | 19.35% | 3 | 9.68% |
| Somewhat disagree | 9 | 29.03% | 9 | 29.03% | 9 | 29.03% | 8 | 26.66% | 9 | 29.03% | 8 | 25.81% |
| Disagree | 6 | 19.35% | 1 | 3.23% | 7 | 22.58% | 3 | 10.00% | 2 | 6.45% | 8 | 25.81% |
| Strongly disagree | 3 | 9.68% | 1 | 3.23% | 6 | 19.35% | 6 | 20.00% | 2 | 6.45% | 7 | 22.58% |

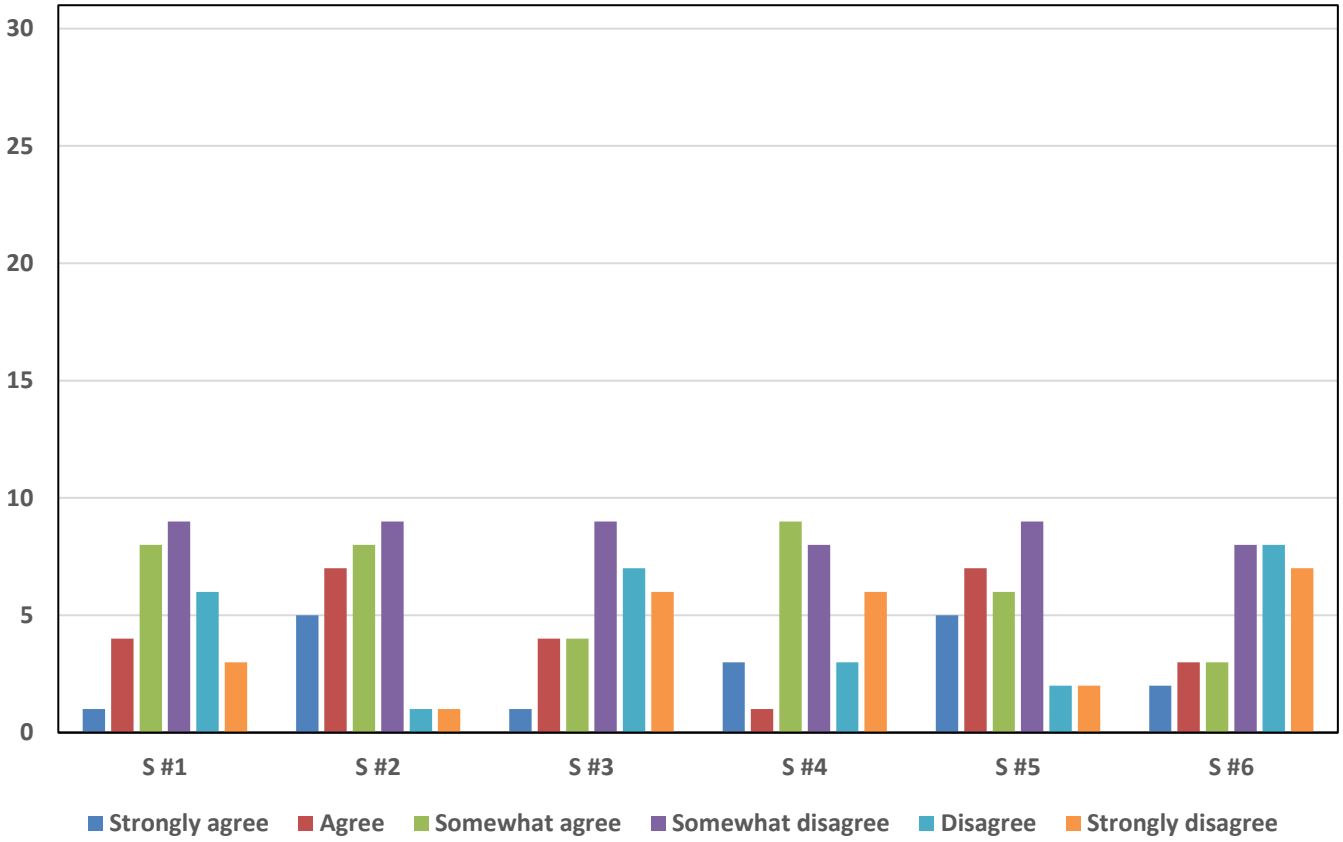



**Figure 4.8. Overall Responses of User Study Participants on GUI-based vs Text-based Modeling.**

### 4.2.5 Parallel Processes and Finite Buffers

As shown in Table 4-13, the fifth block of the questionnaire was comprised of six statements that focused on assessing the ability of the ASAE text-based simulation framework and Arena to model parallel processes and finite capacity buffers effectively. Statements 1, 2, and 3 focused on capturing the opinions of the user study participants about modeling parallel processes. Statements 4, 5, and 6 focused on capturing the opinions of the user study participants about modeling finite capacity buffers.

**Table 4-13. Block 5 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Modeling parallel workstations with ASAE was difficult. |
| 2 | Modeling parallel workstations with Arena was easy. |
| 3 | Compared to Arena, it was more difficult to model parallel workstations with ASAE. |
| 4 | Modeling finite capacity buffers with ASAE was difficult. |
| 5 | Modeling finite capacity buffers with Arena was easy. |
| 6 | Compared to Arena, it was more difficult to model finite capacity buffers with ASAE. |

Table 4-11 shows the responses received for the statements in this block of the user study questionnaire. Figure 4.9 depicts the same results in a graphical format to aid in their interpretation.

The responses received for statement 1 clearly show that the user study participants felt that modeling parallel workstation with the ASAE text-based interface was not difficult, as evidenced by the 81% (i.e., 25 out of 31) of respondents that disagreed with this statement on some level (i.e., “Strongly disagree”, “Somewhat disagree” or “Disagree”). Comparatively, 64.52% of the user study participants agreed on some level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) with statement 2 (i.e., modeling parallel workstation with Arena was easy). When asked if it was harder to model parallel workstations with the ASAE text-based simulation framework when compared to Arena, 74.19% (i.e., 23 out of 31) of the user study participants disagreed on some level. Considering the responses received for statements 1, 2, and 3 collectively, it seems that the user study participants favored the ASAE text-based simulation framework when modeling parallel workstations.

The responses received for statements 4 and 5 are interesting in that the task of modeling finite capacity buffers proved similarly difficult with the ASAE text-based simulation framework and Arena, as evidenced by the percentage of user study participants that agreed on some level with statement 4 (i.e., 12 out of 31, or 38.71%) and disagreed on some level with statement 5 (i.e., 11 out of 31, or 35.48%). However, when responding to statement 6, 21 out of 31 (i.e., 67.74%) of the user study participants disagreed on some level (i.e., “Strongly disagree”, “Somewhat disagree” or “Disagree”) with the premise that modeling finite capacity buffers was more difficult with the ASAE text-based simulation framework than

with Arena. Considering the responses received for statements 4, 5, and 6 collectively, there was no agreement among user study participants about which simulation software approach was easier when modeling finite capacity buffers.

**Table 4-14. Overall Responses of User Study Participants on Parallel Processes and Finite Buffers.**

| | Statement 1 Responses | | Statement 2 Responses | | Statement 3 Responses | | Statement 4 Responses | | Statement 5 Responses | | Statement 6 Responses | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 0 | 0.00% | 2 | 6.45% | 1 | 3.23% | 1 | 3.23% | 2 | 6.45% | 0 | 0.00% |
| Agree | 2 | 6.45% | 12 | 38.71% | 2 | 6.45% | 2 | 6.45% | 11 | 34.48% | 4 | 12.90% |
| Somewhat agree | 4 | 12.90% | 6 | 19.35% | 5 | 16.13% | 9 | 29.03% | 7 | 22.58% | 6 | 19.35% |
| Somewhat disagree | 6 | 19.35% | 4 | 12.90% | 10 | 32.26% | 3 | 9.68% | 2 | 6.45% | 4 | 12.90% |
| Disagree | 15 | 48.93% | 6 | 19.35% | 9 | 29.03% | 9 | 29.03% | 7 | 22.58% | 10 | 32.26% |
| Strongly disagree | 4 | 12.90% | 1 | 3.23% | 4 | 12.90% | 7 | 22.58% | 2 | 6.45% | 7 | 22.58% |

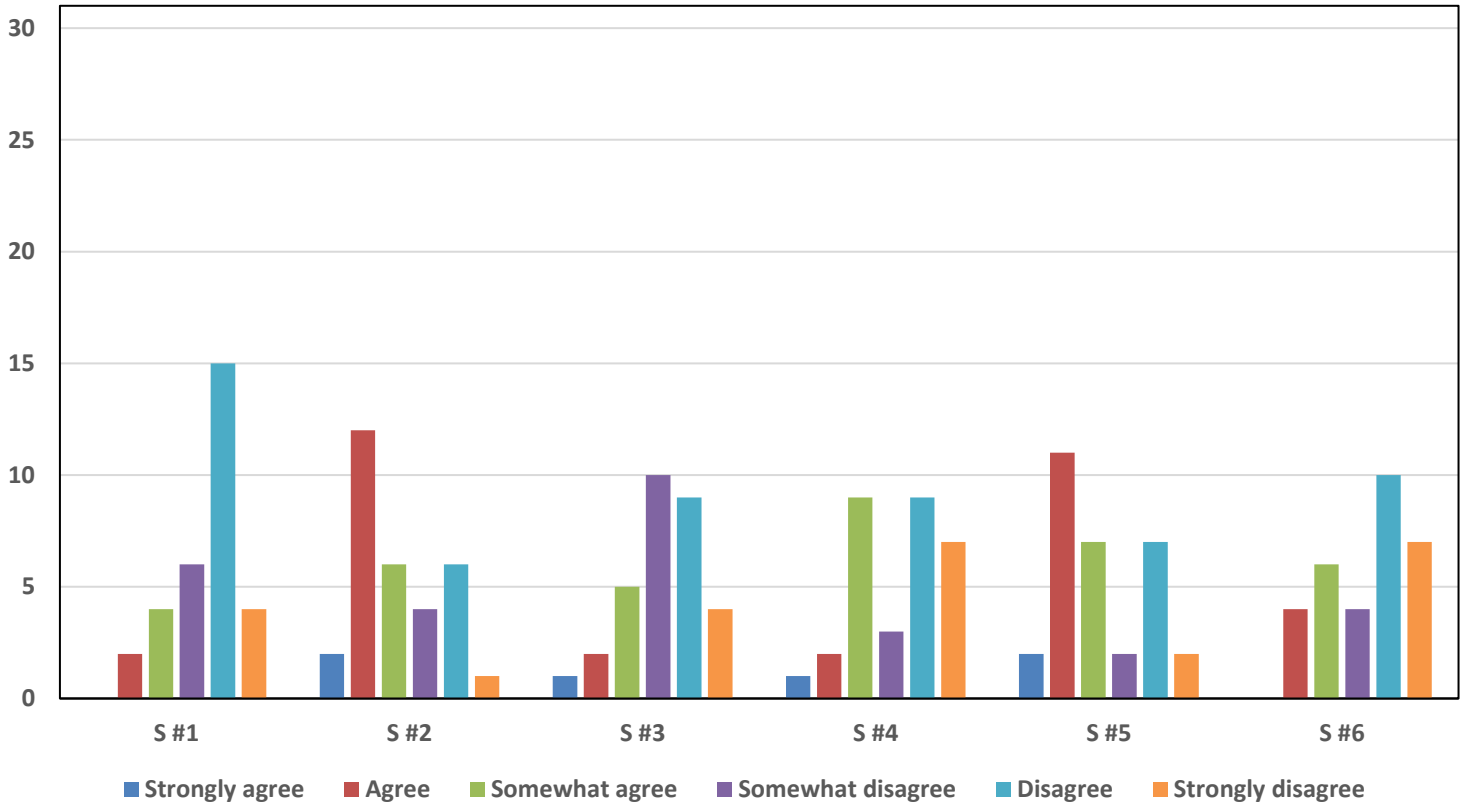



**Figure 4.9. Overall Responses of User Study Participants on Parallel Processes and Finite Buffers.**

### 4.2.6 Understanding the MAS

As shown in Table 4-15, the sixth block of the questionnaire was comprised of six statements. Statements 1 and 2 focused on capturing the opinions of the user study participants about understanding process flow in a MAS. Statements 3 and 4 focused on capturing the opinions of the user study participants about understanding the process characteristics of a MAS. Finally, statements 5 and 6 focused on capturing the opinions of the user study participants about gaining an initial understanding of a MAS.

**Table 4-15. Block 6 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Using Arena's GUI modules allowed me to understand process flow. |
| 2 | Using ASAE's text-based approach allowed me to understand process flow. |
| 3 | Using Arena's GUI modules allowed me to understand the characteristics of each process. |
| 4 | Using ASAE's text-based approach allowed me to understand the characteristics of each process. |
| 5 | Using Arena's GUI modules allowed me to get a better initial understanding of the system. |
| 6 | Using ASAE's text-based approach allowed me to get a better initial understanding of the system. |

Table 4-13 shows the responses received for the statements in this block of the user study questionnaire. Figure 4.10 depicts the same results in a graphical format to aid in their interpretation.

The responses received for statements 1, 3, and 5, which asked user study participants to rate their experience using Arena's GUI-based interface to gain an initial understand of a MAS; understanding process flow in a MAS; and understanding the characteristics of each process in a MAS, were significantly more positive than those for statements 2, 4, and 6, which asked the user study participants to rate their experience when accomplishing the same tasks using the ASAE text-based simulation framework.

Taken collectively, the responses received in this block of the questionnaire clearly show that the user study participants agreed in that Arena's GUI-based interface allows for a better understanding of the general characteristics of a MAS. Additionally, these responses reveal an opportunity for extending the capabilities of the ASAE text-based simulation framework by adding a GUI in the future.

**Table 4-16. Overall Responses of User Study Participants on Understanding the MAS.**

| | Statement 1 Responses | | Statement 2 Responses | | Statement 3 Responses | | Statement 4 Responses | | Statement 5 Responses | | Statement 6 Responses | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 13 | 41.94% | 2 | 6.45% | 10 | 32.26% | 5 | 16.13% | 13 | 41.94% | 3 | 9.68% |
| Agree | 14 | 45.16% | 1 | 3.23% | 13 | 41.94% | 8 | 25.81% | 12 | 38.71% | 1 | 3.23% |
| Somewhat agree | 3 | 9.68% | 8 | 25.81% | 5 | 16.13% | 6 | 19.35% | 5 | 16.15% | 10 | 32.26% |
| Somewhat disagree | 1 | 3.23% | 12 | 38.71% | 3 | 9.68% | 7 | 22.58% | 1 | 3.23% | 10 | 32.26% |
| Disagree | 0 | 0.00% | 5 | 16.13% | 0 | 0.00% | 3 | 9.68% | 0 | 0.00% | 4 | 12.90% |
| Strongly disagree | 0 | 0.00% | 3 | 9.68% | 0 | 0.00% | 2 | 6.45% | 0 | 0.00% | 3 | 9.68% |

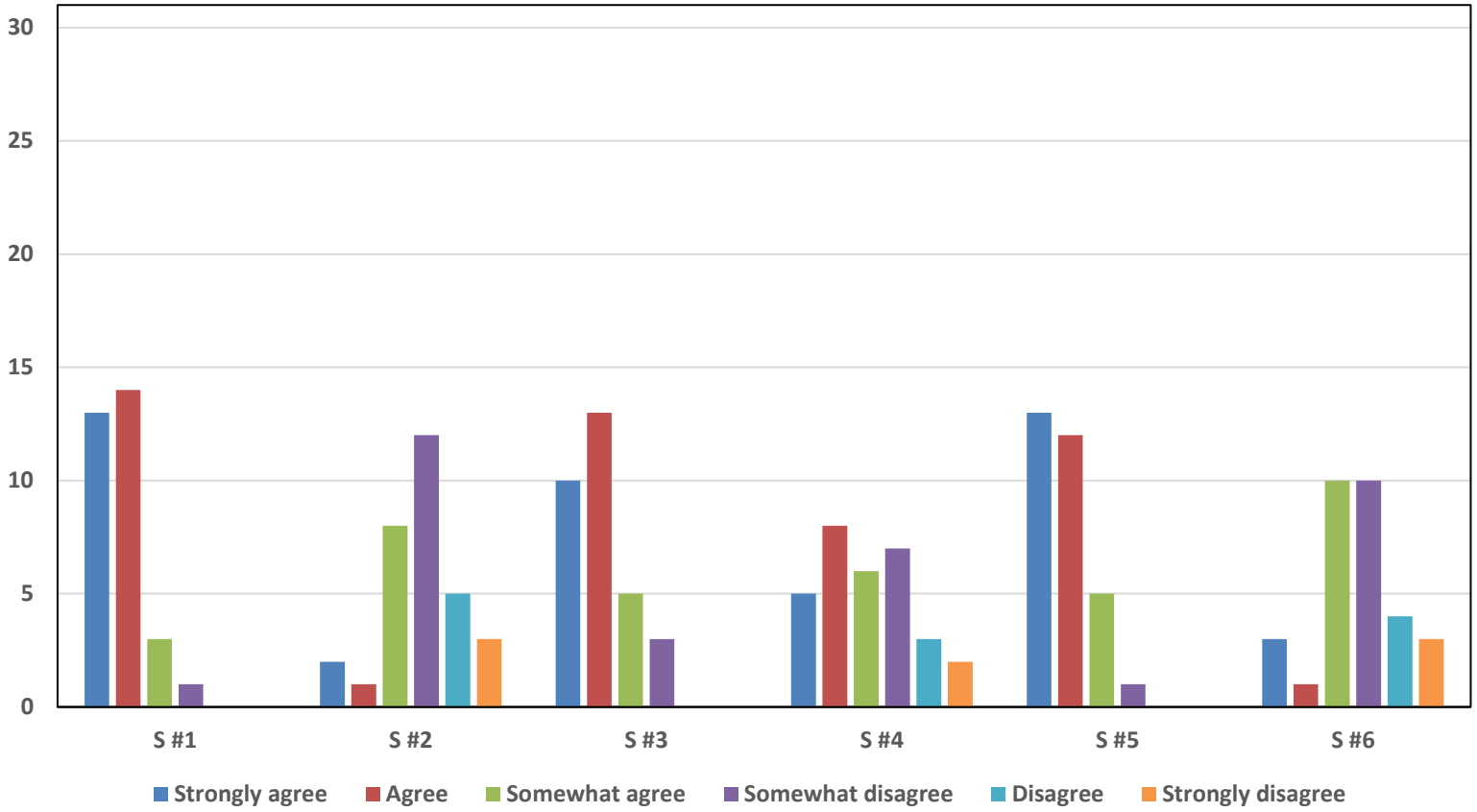



**Figure 4.10. Overall Responses of User Study Participants on Understanding the MAS.**

### 4.2.7 Value of Results

As shown in Table 4-17, the seventh block of the questionnaire was comprised of ten statements that focused on assessing the value of the results provided by the ASAE text-based simulation framework and Arena. More specifically, the statements in this block of the questionnaire were written to capture the opinion of the user study participants about whether or not each simulation software approach allowed them to better understand critical performance characteristics of a MAS, including process flow (i.e., statements 1 and 2), bottlenecks (i.e., statements 3 and 4), throughput (i.e., statements 5 and 6), process times (i.e., statements 7 and 8), and the overall MAS (i.e., statements 9 and 10).

**Table 4-17. Block 7 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | I was able to understand process flow using the Arena results. |
| 2 | I was able to understand process flow using the ASAE results. |
| 3 | I was able to identify potential bottlenecks using the Arena results. |
| 4 | I was able to identify potential bottlenecks using the ASAE results. |
| 5 | I was able to understand the throughput of the system with the Arena results. |
| 6 | I was able to understand the throughput of the system with the ASAE results. |
| 7 | I was able to understand process times with the results provided by Arena. |
| 8 | I was able to understand process times with the results provided by ASAE. |
| 9 | The results provided by Arena allow me to better understand the manufacturing assembly system. |
| 10 | The results provided by ASAE allow me to better understand the manufacturing assembly system. |

Table 4-18 shows the responses received for statements 1, 3, 5, 7, and 9, which correspond to Arena. Table 4-19 shows the responses received for statements 2, 4, 6, 8, and 10, which correspond to the ASAE text-based simulation framework. Figure 4.11 depicts the results for all ten statements in a graphical format to aid in their interpretation.

The responses received for statements 1 and 2 showed a similar pattern. For statement 1, 87.10% (i.e., 27 out of 31) of the study participants agreed on some

level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) that they were able to understand process flow using Arena. For statement 2, 74.19% (i.e., 23 out of 31) of the study participants agreed on some level that they were able to understand process flow using the ASAE text-based simulation framework.

The responses received for statements 3 and 4 were also similar. For statement 3, 90.32% (i.e., 28 out of 31) of the study participants agreed on some level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) that they were able to identify potential bottlenecks using Arena. For statement 4, 77.42% (i.e., 24 out of 31) of the study participants agreed on some level that they were able to identify potential bottlenecks using the ASAE text-based simulation framework.

The responses received for statements 5 and 6 were also similar. For statement 5, 93.55% (i.e., 29 out of 31) of the study participants agreed on some level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) that they were able to understand throughput using Arena. For statement 6, 87.10% (i.e., 27 out of 31) of the study participants agreed on some level that they were able to understand throughput using the ASAE text-based simulation framework.

The responses received for statements 7 and 8 were not only similar, but also the highest (i.e., percentage-wise) in this block of the questionnaire. For statement 7, 93.55% (i.e., 29 out of 31) of the study participants agreed on some level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) that they were able to understand process times using Arena. For statement 8, 90.32% (i.e., 28 out of 31)

of the study participants agreed on some level that they were able to understand process times using the ASAE text-based simulation framework.

The responses received for statements 9 and 10 were also similar. For statement 9, 87.10% (i.e., 27 out of 31) of the study participants agreed on some level (i.e., “Strongly agree”, “Somewhat agree”, or “Agree”) that Arena allowed them to better understand the MAS. For statement 10, 83.87% (i.e., 26 out of 31) of the study participants agreed on some level that the ASAE text-based simulation framework allowed them to better understand the MAS.

Taken collectively, the responses received in this block of the questionnaire clearly show that the user study participants agreed in that the results produced by both Arena and the ASAE text-based simulation framework are useful in better understanding critical performance characteristics of a MAS. It is important to note, however, that the user study participants rated the results produced by Arena more positively than those produced by the ASAE text-based simulation framework.

**Table 4-18. Overall Responses of User Study Participants on Value of Results for the Arena DES Software.**

| | Statement 1 Responses | | Statement 3 Responses | | Statement 5 Responses | | Statement 7 Responses | | Statement 9 Responses | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 11 | 35.48% | 7 | 22.58% | 7 | 22.58% | 9 | 29.03% | 10 | 32.26% |
| Agree | 10 | 32.26% | 15 | 48.39% | 16 | 51.61% | 15 | 48.39% | 11 | 35.48% |
| Somewhat agree | 6 | 19.35% | 6 | 19.35% | 6 | 19.35% | 5 | 16.13% | 6 | 19.35% |
| Somewhat disagree | 2 | 6.45% | 2 | 6.45% | 2 | 6.45% | 1 | 3.23% | 2 | 6.45% |
| Disagree | 2 | 6.45% | 0 | 0.00% | 0 | 0.00% | 1 | 3.23% | 1 | 3.23% |
| Strongly disagree | 0 | 0.00% | 1 | 3.23% | 0 | 0.00% | 0 | 0.00% | 1 | 3.23% |

**Table 4-19. Overall Responses of User Study Participants on Value of Results for the ASAE Text-Based Simulation Framework.**

| | Statement 2 Responses | | Statement 4 Responses | | Statement 6 Responses | | Statement 8 Responses | | Statement 10 Responses | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Likert Scale Point** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** | **Count** | **%** |
| Strongly Agree | 9 | 29.03% | 6 | 19.35% | 9 | 29.03% | 11 | 35.48% | 12 | 38.71% |
| Agree | 11 | 35.48% | 10 | 32.26% | 13 | 41.94% | 12 | 38.71% | 11 | 35.48% |
| Somewhat agree | 3 | 9.68% | 8 | 25.81% | 5 | 16.13% | 5 | 16.13% | 3 | 9.68% |
| Somewhat disagree | 4 | 12.90% | 3 | 9.68% | 3 | 9.68% | 2 | 6.45% | 5 | 16.13% |
| Disagree | 2 | 6.45% | 3 | 9.68% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Strongly disagree | 2 | 6.45% | 1 | 3.23% | 1 | 3.23% | 1 | 3.23% | 0 | 0.00% |

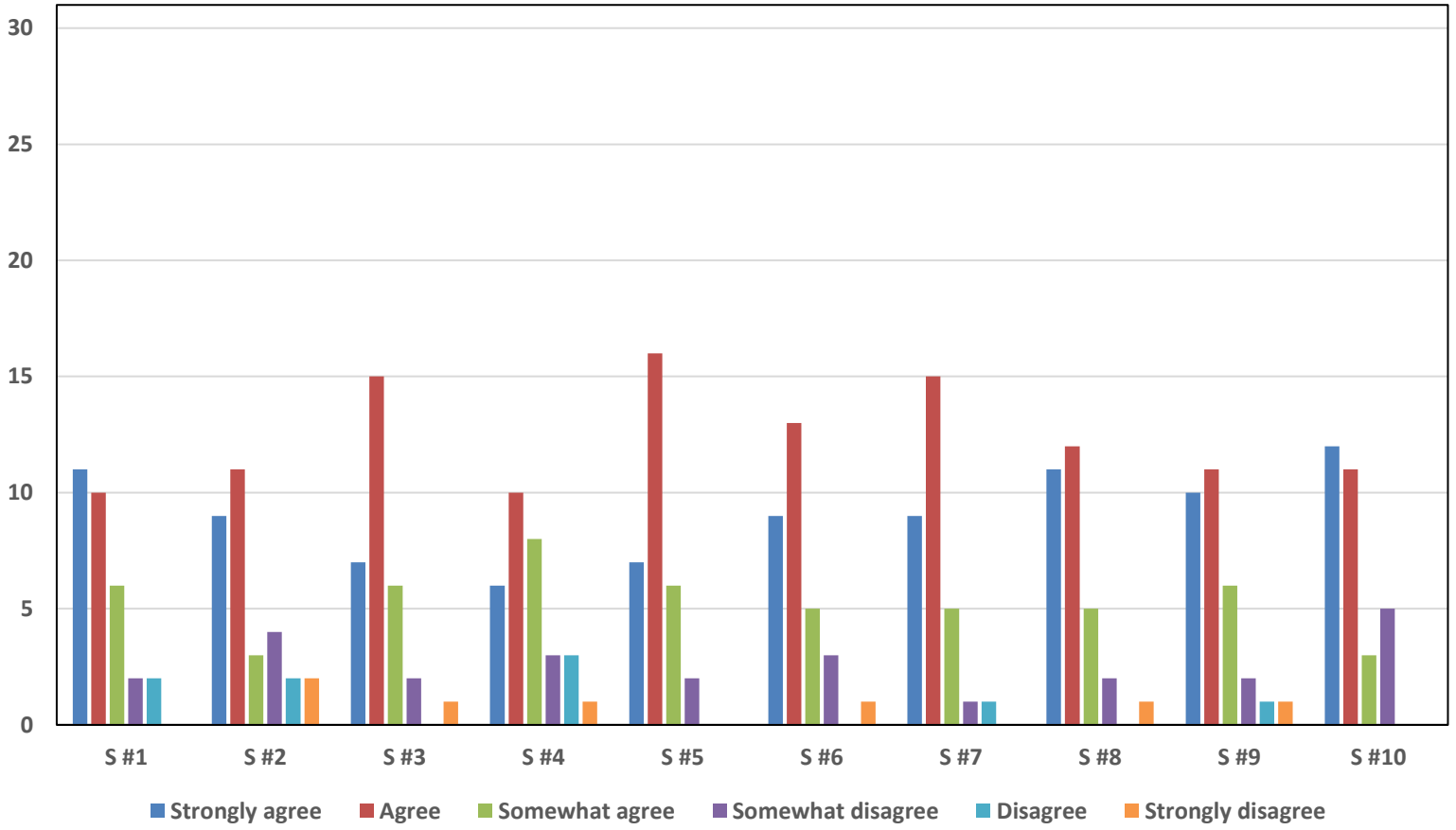



**Figure 4.11. Overall Responses of User Study Participants on Value of Results.**

### 4.2.8 Compare Simulation Platforms

As shown in Table 4-20, the eighth block of the questionnaire was comprised of seven statements that directly compared the ASAE text-based simulation framework against Arena on aspects such as the time needed to create and simulate a MAS, the ability to model parallel workstations, and the ability to model finite buffers, among others.

**Table 4-20. Block 8 Statements of the User Study Questionnaire.**

| Statement # | Statement |
|---|---|
| 1 | Saves more time in creating and simulating a manufacturing assembly system. |
| 2 | Increases the time needed to create and simulate a manufacturing assembly system. |
| 3 | Facilitates the modeling of finite capacity buffers. |
| 4 | Facilitates the modeling of parallel workstations. |
| 5 | The modeling approach is easy to use. |
| 6 | The modeling approach is difficult to understand. |
| 7 | The data generated by the modeling approach provides more insight into the performance of the manufacturing assembly system. |

The user study participants were asked to reflect their opinion about each of the seven statements using the six-point scale depicted in Figure 4.12. The closer a rating was to one of the DES approaches, the more a user study participant was in agreement with the statement relative to the specific DES approach, i.e., a far-left

selection means the statement completely relates to the ASAE text-based simulation framework, whereas a far-right selection means the statement completely relates to the Arena DES software

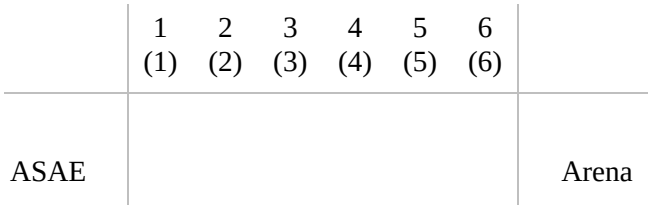



**Figure 4.12. Six-point Scale for Eighth Block of User Study Questionnaire.**

The next seven figures depict the distribution of responses received from the user study participants. Figure 4.13 shows that the user study participants perceived the ASAE text-based simulation framework as being more time efficient than Arena when creating and simulating a MAS (i.e., statement 1). The responses received for statement 2, depicted in Figure 4.14, are in alignment with those observed in statement 1.

Figure 4.15 shows that the user study participants perceived both simulation software approaches as similar when modeling finite capacity buffers (i.e., statement 3). However, the responses received for statement 4 depicted in Figure 4.16 suggest a preference for Arena when modeling parallel workstations.

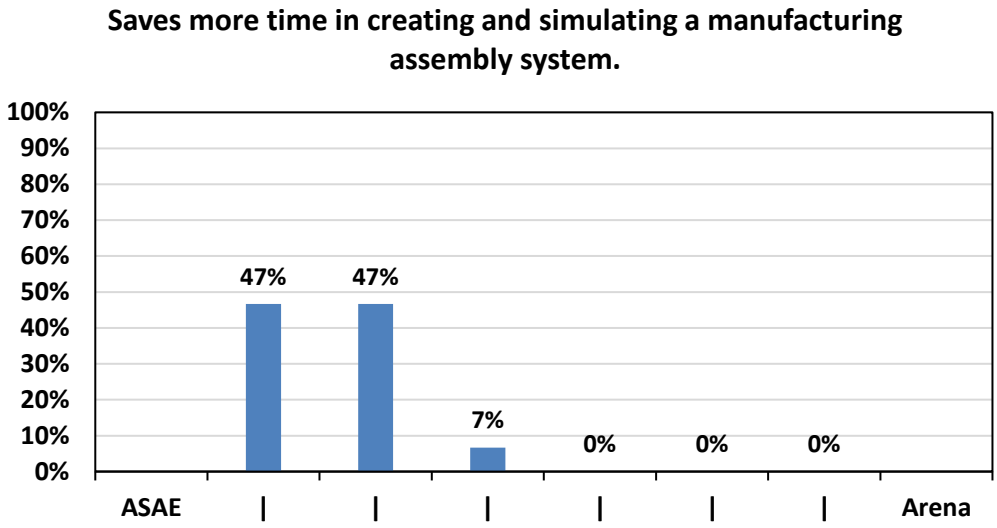



**Figure 4.13. Distribution of Responses for Statement 1.**

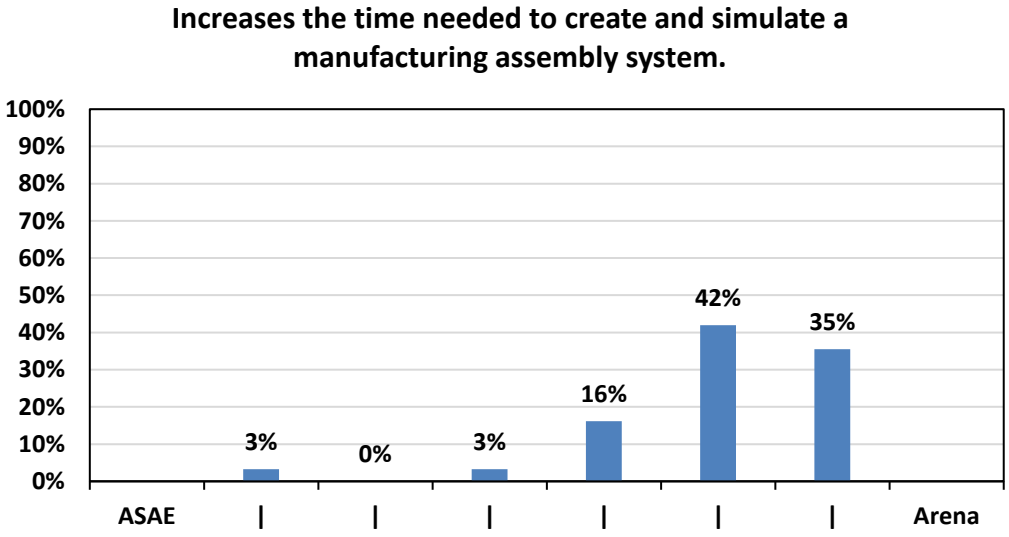



**Figure 4.14. Distribution of Responses for Statement 2.**

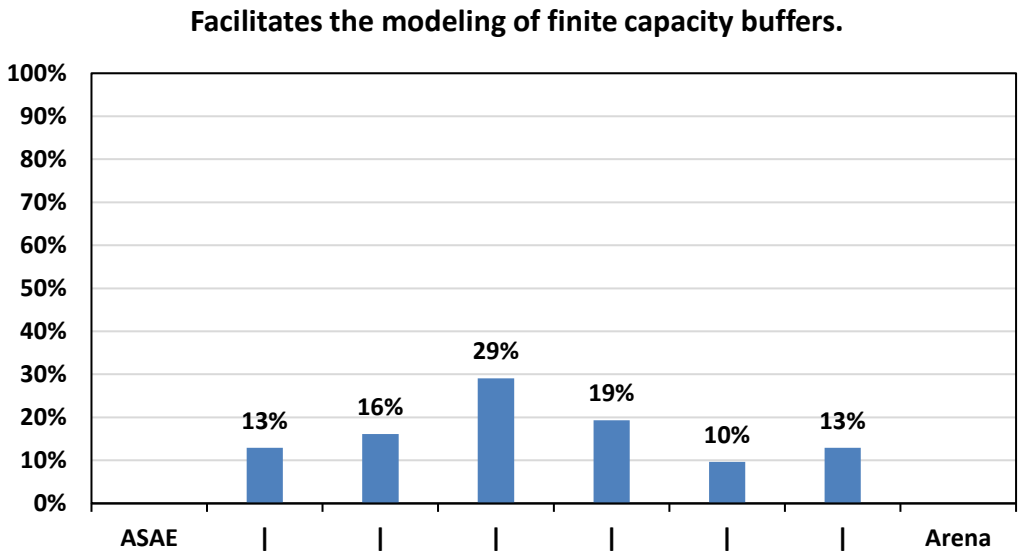



**Figure 4.15. Distribution of Responses for Statement 3.**

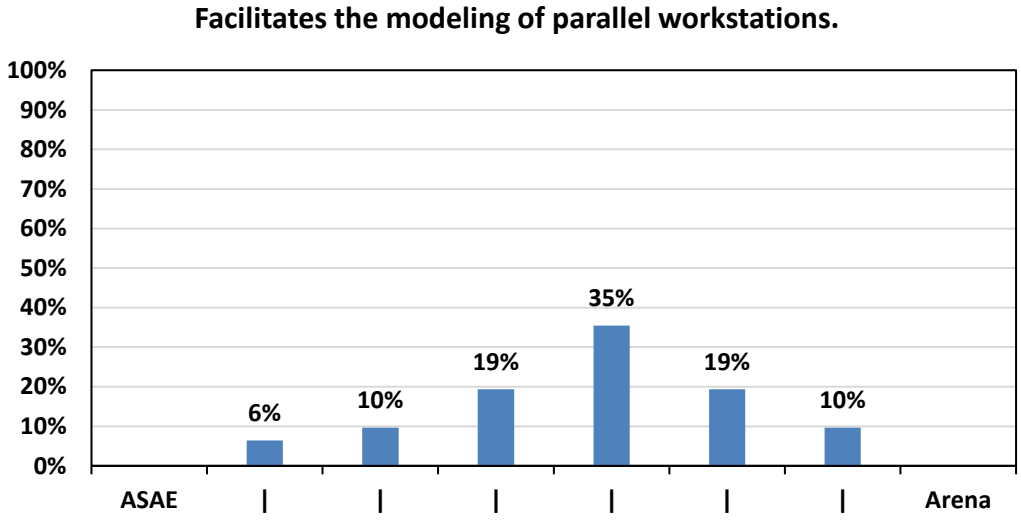



**Figure 4.16. Distribution of Responses for Statement 4.**

Figure 4.17 shows a distribution of responses that support the premise that the ASAE text-based simulation framework is easier to use than Arena when modeling a MAS. In Figure 4.18, almost 70% of the responses provided by the user study participants concentrate in the middle of the rating scale, which suggest that neither simulation software approach was perceived as more difficult to understand than the other.

Finally, the responses depicted in Figure 4.19 show an almost uniform distribution which suggests that the results provided by the ASAE text-based simulation framework and Arena were perceived by the user study participants as equivalent when providing insight into the performance of a MAS.

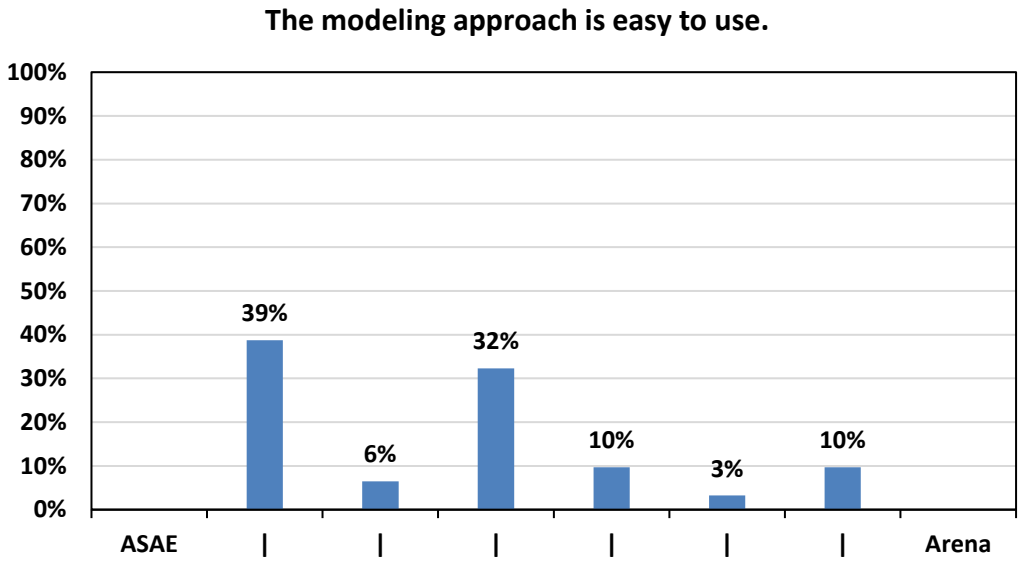



**Figure 4.17. Distribution of Responses for Statement 5.**

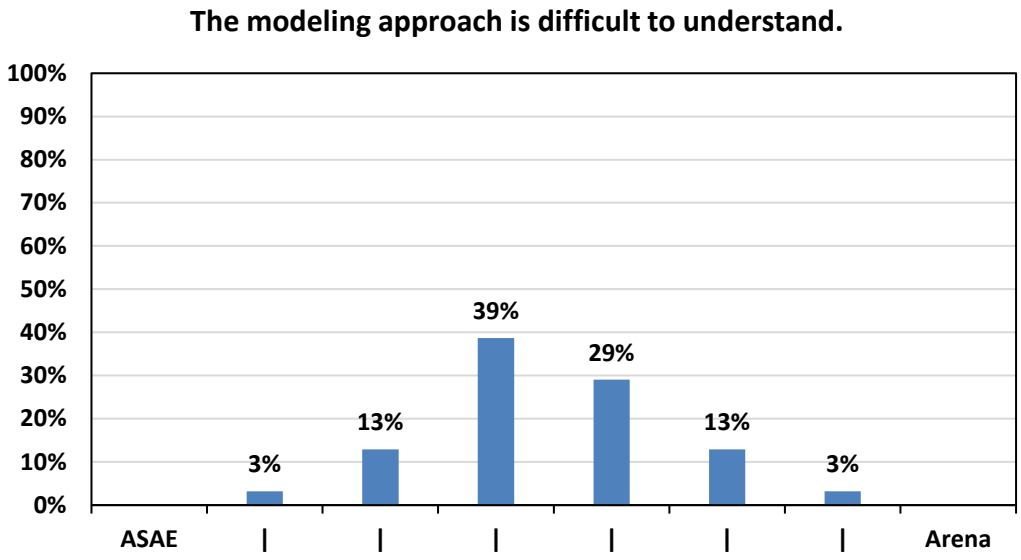



**Figure 4.18. Distribution of Responses for Statement 6.**

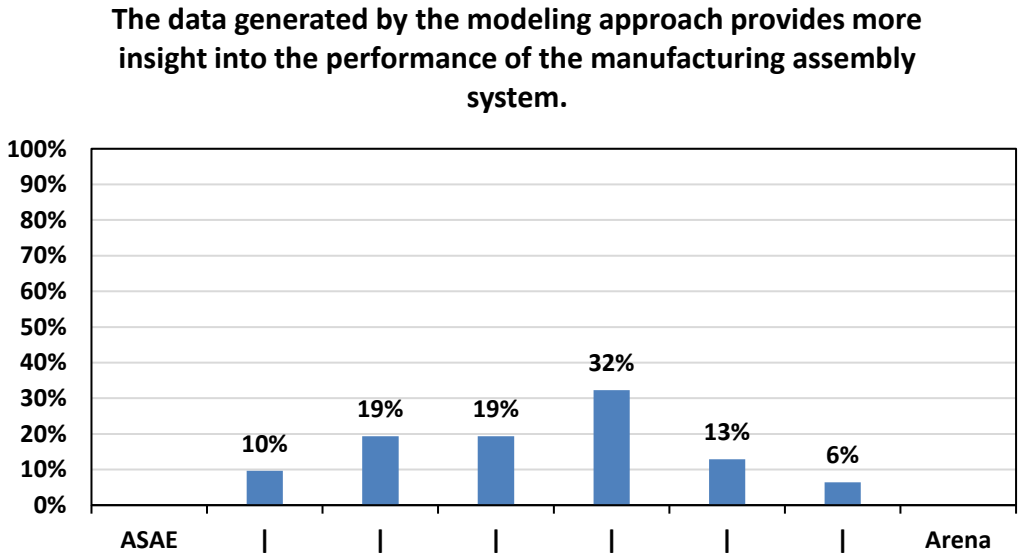



**Figure 4.19. Distribution of Responses for Statement 7.**

### 4.2.9 Preference

In the last block of the questionnaire, the user study participants were asked to select the modeling approach they would rather use to create and simulate a MAS. As Figure 4.20 illustrates, 67.74% (i.e., 21 out of 31) of the user study participants chose the ASAE text-based simulation framework over Arena.

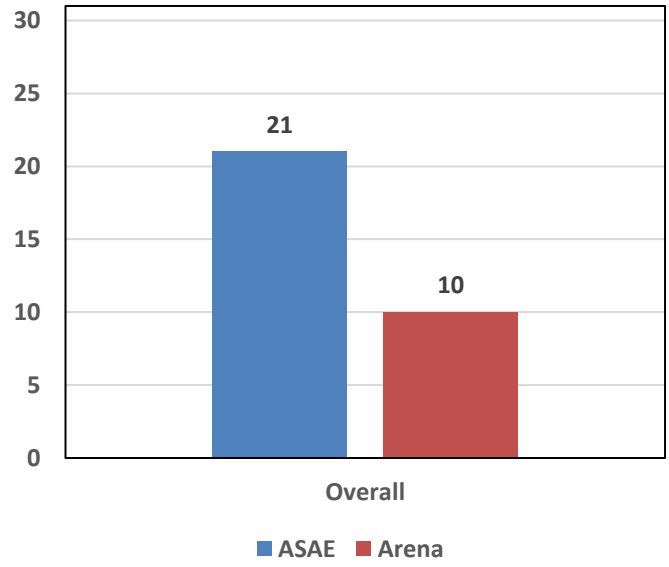



**Figure 4.20. Preference of User Study Participants.**

As described in Section 4.2.1, the user study participants were asked in the first block of the questionnaire to rate their ability to develop simulation models as either "Excellent", "Good", "Average", or "Poor" after they had completed the pre-study training exercises using both simulation software approaches. To complement the responses received for the participant's preference depicted in

Figure 4.20, a filtering function was applied to the questionnaire results stored in Qualtrics to determine which simulation software approach had been selected by the user study participants in each of the four categories of ability. The results of applying this filter are depicted in Figure 4.21.

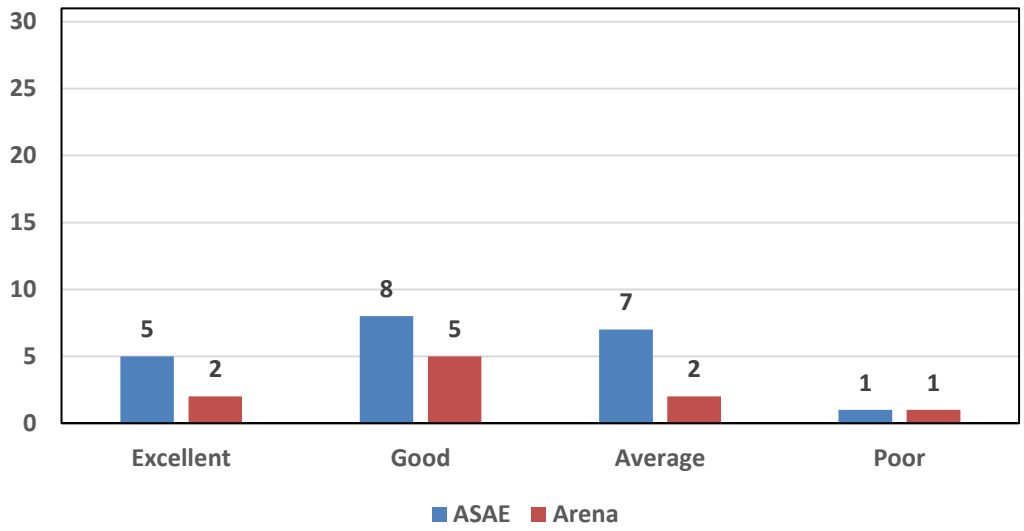



**Figure 4.21. DES Approach Preference by User Ability Category**

The responses in Figure 4.21 show that in every ability category other than "Poor", the ASAE text-based simulation framework was preferred over Arena. These responses suggest that the user study participants prefer the ASAE text-based simulation framework over Arena.

### 4.2.10 Synthesis of Questionnaire Results

The results of the questionnaire offer strong evidence that the modeling capabilities and user preference are important factors to determine the value and usability of the ASAE text-based simulation framework. However, the results when building and simulating a MAS with a GUI-based versus a text-based approach or with regards to the ability to understand a MAS indicate that there is no perceived difference between the ASAE text-based simulation framework and Arena. With study participants possessing varying skills and skill levels, it is inferred that users with any amount of experience in simulation technologies can utilize the capabilities offered by the ASAE text-based simulation framework under simple simulation environments.

In summary, the results of the questionnaire indicate that the ASAE text-based simulation framework provides value for users needing to incorporate a simulation modeling process, but it is unclear whether the ASAE text-based simulation framework provides a comprehensive understanding of a MAS.

# 5 CONCLUSIONS AND OPPORTUNITIES FOR FUTURE WORK

This chapter presents the conclusions and opportunities for future work of this research project. The conclusions are presented in Section 5.1, whereas Section 5.2 outlines the opportunities for future work.

## 5.1 CONCLUSIONS

The acquisition and licensing costs, the level of expertise required, and the time and resources needed to maintain models of manufacturing assembly systems (MASs) are significant barriers for the widespread use of discrete event simulation (DES) software packages. Therefore, the objective of this research was to develop a methodology to automate the process of creating, simulating, and analyzing a MAS. The main contribution of this research was the development of an automated, text-based simulation framework referred to as the Automated Simulation Analysis Engine (ASAE). The proposed ASAE text-based simulation framework includes the following features:

- A text-based modeling approach,
- Automated data collection, and
- Automated simulation and analysis of a MAS.

Several tests were conducted to ensure that the design and implementation of the ASAE text-based simulation framework were done properly and correctly.

First, the degree of randomness of the simulation engine of the ASAE text-based simulation framework was compared to that of Arena, a widely used DES software package. The results of this test showed that the two simulation systems performed as expected, and confirmed the randomness of the simulation engine of the ASAE text-based simulation framework.

In a second test, 20 jobs with constant time were mapped manually for a MAS Type II consisting of three processes and two finite buffers. The results of the manual mapping process were identical to those obtained by simulating 20 jobs with constant time for the same MAS Type II with the ASAE text-based simulation framework and Arena, which validated the complex set of rules applied by the ASAE text-based simulation engine to dictate how events are scheduled and processed based on various dependencies and times. An additional test was conducted in which 100 jobs with constant time were simulated with the ASAE text-based simulation framework and Arena to further validate the logic employed by the simulation engine of the ASAE text-based simulation framework.

Finally, a user study was designed and conducted to collect feedback from unbiased users (with at least some exposure to simulation technologies) about the usability and potential value and of the ASAE text-based simulation framework. As part of the user study, 31 subjects were administered a questionnaire *after* they had completed a set of pre-study training exercises on the Arena DES software and the ASAE text-based simulation framework. The results of the user study suggest that the ASAE text-based simulation framework has significant potential in saving

time and reducing the project timeline of simulation-based projects involved in the simulation of MASs with finite buffer resources and parallel processes.

The ASAE text-based simulation framework was developed as open source software and runs in many different platforms. Through a web-based model creation interface, the ASAE text-based simulation framework automates the data collection and data analysis processes to facilitate the quick characterization of a MAS with parallel workstations and finite buffer resources. The main benefits of the ASAE text-based simulation framework are as follows:

- It may reduce the time needed to execute simulation tasks of MASs with finite buffer resources and parallel processes.
- It may reduce the budget needed to start and finish simulation-based projects that focus on MASs with finite buffer resources and parallel processes.
- It may reduce the amount of simulation expertise required to employ a simulation-based framework when considering MASs with finite buffer resources and parallel workstations.
- Finally, it may facilitate the quick modeling and iteration of simulation models.

While the ASAE text-based simulation framework has proven to be effective in several areas, the feedback received from the user study also revealed some limitations, including:

- Using the ASAE text-based simulation framework makes it more difficult to gain an understanding of a MAS through modeling alone.
- The ASAE text-based simulation framework lacks a dynamic visual interface to indicate what is taking place during the execution of a simulation.
- The current design has limitations in the type of MAS constructs that can be modeled correctly.
- ASAE text-based simulation framework does not account for resources and resource utilization.
- The current design of the ASAE text-based simulation framework does not allow processes to share buffers.

## 5.2 OPPORTUNITIES FOR FUTURE WORK

Based on the work presented and the feedback received through the user study, there are several opportunities for future work that can extend this body of research, including:

- The creation of dynamic visual extensions for simulation approaches based on simplified text descriptions.
- The addition of simplified Graphical User Interface (GUI) interfaces that eliminate the requirement for expertise in simulation technologies.
- The incorporation of complex resources into the text-based model definition.

- The development of a front-end GUI to bring the ASAE text-based simulation framework off the command line.

# 6 BIBLIOGRAPHY

Aalst, W. V. (2016). Process Mining Data Science in Action. Berlin: Springer Berlin.

Aalst, W. V., Weijters, T., & Maruster, L. (2004). Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16(9), 1128-1142.

Aguirre, S., Parra, C., & Alvarado, J. (2013). Combination of Process Mining and Simulation Techniques for Business Process Redesign: A Methodological Approach. Lecture Notes in Business Information Processing Data-Driven Process Discovery and Analysis, 24-43.

Ahn, S., Dunston, P. S., Kandil, A., & Martinez, J. C. (2015). Process Mining Technique for Automated Simulation Model Generation Using Activity Log Data. Computing in Civil Engineering 2015.

Alaskari, O., Ahmad, M. M., & Cuenca, R. P. (2014). Critical success factors for Lean tools and ERP systems implementation in manufacturing SMEs. International Journal of Lean Enterprise Research, 1(2), 183.

Alhuraish, I., Robledo, C., & Kobi, A. (2016). Assessment of Lean Manufacturing and Six Sigma operation with Decision Making Based on the Analytic Hierarchy Process. IFAC-PapersOnLine, 49(12), 59-64.

Aziz, A., Jarrahi, F., & Abdul-Kader, W. (2010). Modeling and Performance Evaluation of a Series-parallel Flow Line System with Finite Buffers. INFOR: Information Systems and Operational Research, 48(2), 103-120.

Barlas, P., Heavey, C., & Dagkakis, G. (2015). An Open Source Tool For Automated Input Data In Simulation. *Internation Journal of Simulation Modeling*.

Bergmann, S., & Strassburger, S. (2010). Challenges for the Automatic Generation of Simulation Models for Production Systems.

Byrne, N., Liston, P., Geraghty, J., & Young, P. (2012). The Potential Role of Open Source Discrete Event Simulation Software in the Manufacturing Sector.

Fournier-Viger, P., Nkambou, R., & Tseng, V. S. (2011). RuleGrowth. Proceedings of the 2011 ACM Symposium on Applied Computing - SAC 11.

Gronniger, H., Krahn, H., Rumpe, B., Schindler, M., & Volkel, S. (2007). Text-based Modeling.

Ham, W. K., & Park, S. C. (2014). A framework for the continuous performance improvement of manned assembly lines. International Journal of Production Research,52(18), 5432-5450.

Haraszko, C., Nemeth, I., & Baldwin, J. (2013). DES Configurators for Rapid Prototyping of Manufacturing systems. *International conference on Innovative Technologies*.

Heavey, C., & Robin, S. (2014). Development of an Open-Source Discrete Event Simulation Cloud Enabled Platform.

Hughes, R., Scott, R., & Ridgway, K. (2013). Automatic simulation model generation for supporting facility planning in SMEs.

Kelton, W. D., Sadowski, R. P., & Zupick, N. B. (2010). Simulation with Arena. New York, NY: McGraw-Hill Education.

Köksal, G., Batmaz, I., & Testik, M. C. (2011). A review of data mining applications for quality improvement in manufacturing industry. Expert Systems with Applications, 38(10), 13448-13467.

Lee, Y. T., Riddick, F. H., & Johansson, B. I. (2011). Core Manufacturing Simulation Data - a manufacturing simulation integration standard: overview and case studies. International Journal Of Computer Integrated Manufacturing, 24(8), 689-709.

Li, L., Chang, Q., Ni, J., Xiao, G., & Biller, S. (2007). Bottleneck Detection of Manufacturing Systems Using Data Driven Method. 2007 IEEE International Symposium on Assembly and Manufacturing.

Mourtzis, Doukas, & Bernidaki. (2014). Simulation in Manufacturing: Review and Challenges. *Procedia CIRP*., 213-229.

Netjes, M., Vanderfeesten, I., & Reijers, H. A. (2006). "Intelligent" Tools for Workflow Process Redesign: A Research Agenda. Business Process Management Workshops Lecture Notes in Computer Science, 444-453.

Pillai, V. M., & Chandrasekharan, M. (2008). An absorbing Markov chain model for production systems with rework and scrapping. Computers & Industrial Engineering, 55(3), 695-706.

Rossetti, M.D. (2008). Java Simulation Library (JSL): an open-source object-oriented library for discrete-event simulation in Java. *IJSPM, 4*, 69-87.

Rozinat, A., Jong, I. D., Gunther, C., & Aalst, W. V. (2009). Process Mining Applied to the Test Process of Wafer Scanners in ASML. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),39(4), 474-479.

Rozinat, A., Mans, R. S., Song, M., & W. M. P. Van Der Aalst. (2007). Discovering colored Petri nets from event logs. International Journal on Software Tools for Technology Transfer, 10(1), 57-74.

Rozinat, A., Mans, R., Song, M., & Aalst, W. V. (2009). Discovering simulation models. Information Systems,34(3), 305-327.

Senanayake, C. D., & Subramaniam, V. (2011). Analysis of a two-stage, flexible production system with unreliable machines, finite buffers and non-negligible setups. Flexible Services and Manufacturing Journal, 25(3), 414-442.

Van Der Aalst, W. (2012). Process mining: Overview and opportunities. ACM Transactions on Management Information Systems (TMIS), 3(2), 7.

Wang, J., Wong, R. K., Ding, J., Guo, Q., & Wen, L. (2013). Efficient Selection of Process Mining Algorithms. IEEE Transactions on Services Computing, 6(4), 484-496.

Wedel, M., Hacht, M. V., Hieber, R., Metternich, J., & Abele, E. (2015). Real-time Bottleneck Detection and Prediction to Prioritize Fault Repair in Interlinked Production Lines. Procedia CIRP, 37, 140-145.

Ylipaa, T., & Bolmsjo, G. (2005). Reducing bottle-necks in a manufacturing system with automatic data collection and discrete event simulation. *Journal of Manufacturing Technology Management*.

Zheng, Li, Liang Tang, Tao Li, Bing Duan, Ming Lei, Pengnian Wang, Chunqiu Zeng, Lei Li, Yexi Jiang, Wei Xue, Jingxuan Li, Chao Shen, Wubai Zhou, and Hongtai Li (2014). "Applying data mining techniques to address critical process optimization needs in advanced manufacturing." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14 (2014): n. pag. Web.

# 7 APPENDICES

# APPENDIX A

## Arena VBA Code for MAS with Parallel Workstations and Finite Buffers

```
Option Explicit

Dim oSiman As Arena.SIMAN, jobIDAttindx As Long, startTimeindex1 As
Long, startTimeindex2 As Long, startTimeindex3 As Long
Dim nNextRow As Long

Dim oExcelApp As Excel.Application, oWorkbook As Excel.Workbook,
oWorksheet As Excel.Worksheet

Private Sub ModelLogic_RunBeginSimulation()
nNextRow = 2
Set oSiman = ThisDocument.Model.SIMAN
jobIDAttindx = oSiman.SymbolNumber("jobID")
startTimeindex1 = oSiman.SymbolNumber("Process1Enter")
startTimeindex2 = oSiman.SymbolNumber("Process2Enter")
startTimeindex3 = oSiman.SymbolNumber("Process3Enter")

Set oExcelApp = CreateObject("Excel.Application")
oExcelApp.Visible = True
oExcelApp.SheetsInNewWorkbook = 1
Set oWorkbook = oExcelApp.Workbooks.Add

Set oWorksheet = oWorkbook.ActiveSheet
With oWorksheet
   .Name = "Event Log"
   .Cells(1, 1).value = "JobID"
   .Cells(1, 2).value = "ProcessID"
   .Cells(1, 3).value = "StartTime"
   .Cells(1, 4).value = "EndTime"
   .Cells(1, 5).value = "Resource"

   End With
End Sub


Private Sub VBA_Block_1_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
```

```
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex1)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
   .Cells(nNextRow, 1).value = jid & "A"
   .Cells(nNextRow, 2).value = "Process 1"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = simTime
   .Cells(nNextRow, 5).value = "Worker 1"
End With

nNextRow = nNextRow + 1


End Sub

Private Sub VBA_Block_2_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex2)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

With oWorksheet
   .Cells(nNextRow, 1).value = jid & "B"
   .Cells(nNextRow, 2).value = "Process 2"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = simTime
   .Cells(nNextRow, 5).value = "Worker 2"
End With

nNextRow = nNextRow + 1
End Sub

Private Sub VBA_Block_3_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex3)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)
```

```
With oWorksheet
   .Cells(nNextRow, 1).value = jid & "C"
   .Cells(nNextRow, 2).value = "Process 3"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = simTime
   .Cells(nNextRow, 5).value = "Worker 3"
End With

nNextRow = nNextRow + 1
End Sub
```

# APPENDIX B

## Arena VBA Code for Complex MAS

```
Option Explicit

Dim oSiman As Arena.SIMAN, jobIDAttindx As Long, startTimeindex1 As
Long, startTimeindex2 As Long, startTimeindex3 As Long, startTimeindex4 As
Long, startTimeindex5 As Long, startTimeindex6 As Long,
reworkStartTimeIndex As Long

Dim reworkFlagIndex As Long


Dim nNextRow As Long
Dim nNextRowOut As Long

Dim oExcelApp As Excel.Application, oWorkbook As Excel.Workbook,
oWorksheet As Excel.Worksheet

Private Sub ModelLogic_RunBeginSimulation()
nNextRow = 2
nNextRowOut = 2
Set oSiman = ThisDocument.Model.SIMAN
jobIDAttindx = oSiman.SymbolNumber("jobID")
startTimeindex1 = oSiman.SymbolNumber("Process1Enter")
startTimeindex2 = oSiman.SymbolNumber("Process2Enter")
startTimeindex3 = oSiman.SymbolNumber("Process3Enter")
startTimeindex4 = oSiman.SymbolNumber("Process4Enter")
startTimeindex5 = oSiman.SymbolNumber("Process5Enter")
startTimeindex6 = oSiman.SymbolNumber("Process6Enter")
reworkStartTimeIndex = oSiman.SymbolNumber("ReworkEnter")
reworkFlagIndex = oSiman.SymbolNumber("reworkDone")

Set oExcelApp = CreateObject("Excel.Application")
oExcelApp.Visible = True
oExcelApp.SheetsInNewWorkbook = 1
Set oWorkbook = oExcelApp.Workbooks.Add

Set oWorksheet = oWorkbook.ActiveSheet
With oWorksheet
   .Name = "Event Log"
   .Cells(1, 1).value = "JobIDEnter"
```

```
    .Cells(1, 2).value = "ProcessID_Enter"
    .Cells(1, 3).value = "StartTime"
    .Cells(1, 4).value = "Resource"
    .Cells(1, 6).value = "JobIDExit"
    .Cells(1, 7).value = "ProcessID_Exit"
    .Cells(1, 8).value = "EndTime"

    End With
End Sub


Private Sub VBA_Block_1_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex1)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
    .Cells(nNextRow, 1).value = jid
    .Cells(nNextRow, 2).value = "A"
    .Cells(nNextRow, 3).value = startTime
    .Cells(nNextRow, 4).value = "Worker 1"
    .Cells(nNextRow, 6).value = jid
    .Cells(nNextRow, 7).value = "A"
    .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1


End Sub

Private Sub VBA_Block_2_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex2)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

With oWorksheet
```

```
    .Cells(nNextRow, 1).value = jid
    .Cells(nNextRow, 2).value = "B"
    .Cells(nNextRow, 3).value = startTime
    .Cells(nNextRow, 4).value = "Worker 2"
    .Cells(nNextRow, 6).value = jid
    .Cells(nNextRow, 7).value = "B"
    .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1
End Sub

Private Sub VBA_Block_3_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex3)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

With oWorksheet
    .Cells(nNextRow, 1).value = jid
    .Cells(nNextRow, 2).value = "AC"
    .Cells(nNextRow, 3).value = startTime
    .Cells(nNextRow, 4).value = "Worker 3"
    .Cells(nNextRow, 6).value = jid
    .Cells(nNextRow, 7).value = "C"
    .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1

With oWorksheet
    .Cells(nNextRow, 1).value = jid
    .Cells(nNextRow, 2).value = "BC"
    .Cells(nNextRow, 3).value = startTime
    .Cells(nNextRow, 4).value = "Worker 3"
    .Cells(nNextRow, 6).value = "x"
    .Cells(nNextRow, 7).value = "x"
    .Cells(nNextRow, 8).value = "x"

End With
```

```
nNextRow = nNextRow + 1
With oWorksheet
   .Cells(nNextRow, 1).value = jid
   .Cells(nNextRow, 2).value = "DC"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = "Worker 3"
   .Cells(nNextRow, 6).value = "x"
   .Cells(nNextRow, 7).value = "x"
   .Cells(nNextRow, 8).value = "x"

End With

nNextRow = nNextRow + 1


End Sub

Private Sub VBA_Block_4_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex4)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
   .Cells(nNextRow, 1).value = jid
   .Cells(nNextRow, 2).value = "D"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = "Worker 4"
   .Cells(nNextRow, 6).value = jid
   .Cells(nNextRow, 7).value = "D"
   .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1
End Sub

Private Sub VBA_Block_5_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
```

```
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex5)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
   .Cells(nNextRow, 1).value = jid
   .Cells(nNextRow, 2).value = "E"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = "Worker 5"
   .Cells(nNextRow, 6).value = jid
   .Cells(nNextRow, 7).value = "E"
   .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1
End Sub

Private Sub VBA_Block_6_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex6)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

With oWorksheet
   .Cells(nNextRow, 1).value = jid
   .Cells(nNextRow, 2).value = "CF"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = "Worker 6"
   .Cells(nNextRow, 6).value = jid
   .Cells(nNextRow, 7).value = "F"
   .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1

With oWorksheet
   .Cells(nNextRow, 1).value = jid
   .Cells(nNextRow, 2).value = "EF"
   .Cells(nNextRow, 3).value = startTime
```

```
    .Cells(nNextRow, 4).value = "Worker 6"
    .Cells(nNextRow, 6).value = "x"
    .Cells(nNextRow, 7).value = "x"
    .Cells(nNextRow, 8).value = "x"

End With

nNextRow = nNextRow + 1
End Sub

Private Sub VBA_Block_7_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, reworkStartTimeIndex)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
    .Cells(nNextRow, 1).value = jid
    .Cells(nNextRow, 2).value = "AR"
    .Cells(nNextRow, 3).value = startTime
    .Cells(nNextRow, 4).value = "Rework 1"
    .Cells(nNextRow, 6).value = jid
    .Cells(nNextRow, 7).value = "R"
    .Cells(nNextRow, 8).value = simTime

End With

nNextRow = nNextRow + 1
End Sub
```

# APPENDIX C

## Arena VBA Code for Simple Linear MAS

```
Option Explicit

Dim oSiman As Arena.SIMAN, jobIDAttindx As Long, startTimeindex1 As
Long, startTimeindex2 As Long, startTimeindex3 As Long
Dim nNextRow As Long

Dim oExcelApp As Excel.Application, oWorkbook As Excel.Workbook,
oWorksheet As Excel.Worksheet

Private Sub ModelLogic_RunBeginSimulation()
nNextRow = 2
Set oSiman = ThisDocument.Model.SIMAN
jobIDAttindx = oSiman.SymbolNumber("jobID")
startTimeindex1 = oSiman.SymbolNumber("Process1Enter")
startTimeindex2 = oSiman.SymbolNumber("Process2Enter")
startTimeindex3 = oSiman.SymbolNumber("Process3Enter")

Set oExcelApp = CreateObject("Excel.Application")
oExcelApp.Visible = True
oExcelApp.SheetsInNewWorkbook = 1
Set oWorkbook = oExcelApp.Workbooks.Add

Set oWorksheet = oWorkbook.ActiveSheet
With oWorksheet
   .Name = "Event Log"
   .Cells(1, 1).value = "JobID"
   .Cells(1, 2).value = "ProcessID"
   .Cells(1, 3).value = "StartTime"
   .Cells(1, 4).value = "EndTime"
   .Cells(1, 5).value = "Resource"

   End With
End Sub


Private Sub VBA_Block_1_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
```

```
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex1)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)


With oWorksheet
   .Cells(nNextRow, 1).value = jid & "A"
   .Cells(nNextRow, 2).value = "Process 1"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = simTime
   .Cells(nNextRow, 5).value = "Worker 1"
End With

nNextRow = nNextRow + 1


End Sub


Private Sub VBA_Block_3_Fire()
Dim jid As Long
Dim simTime As Double, startTime As Double
startTime = oSiman.EntityAttribute(oSiman.ActiveEntity, startTimeindex3)
simTime = oSiman.RunCurrentTime
jid = oSiman.EntityAttribute(oSiman.ActiveEntity, jobIDAttindx)

With oWorksheet
   .Cells(nNextRow, 1).value = jid & "C"
   .Cells(nNextRow, 2).value = "Process 3"
   .Cells(nNextRow, 3).value = startTime
   .Cells(nNextRow, 4).value = simTime
   .Cells(nNextRow, 5).value = "Worker 3"
End With

nNextRow = nNextRow + 1
End Sub
```

# APPENDIX D

## Consent Form

# Usability Study of a New Text-based Simulation Modeling Approach Consent Form

Welcome to the research study! We are interested in understanding the advantages of using a new text-based discrete event simulation (DES) modeling approach called Automated Simulation Analysis Engine (ASAE) to create and simulate manufacturing assembly systems. As a study participant, you will be presented with information relevant to the use of ASAE and Arena and then will complete two modeling exercises followed by an online questionnaire to collect usability data. The data collected within this study will be used and published in support of the student researcher's M.S. thesis. All study data will be kept confidential.

The study should take approximately one hour to complete, and you will receive $10.00 in cash for your participation. Your participation in this research is voluntary. Your decision to take part or not take part in this study will not affect your grades, your relationship with your professors, or standing in the University.

You have the right to withdraw at any point during the study, for any reason, and without any prejudice. If you would like to contact the study's Principal Investigator to discuss this research, please e-mail David.Porter@oregonstate.edu. If you have questions about your rights or welfare as a participant, please contact the Oregon State University Human Research Protection Program (HRPP) office, at (541) 737-8008 or by email at IRB@oregonstate.edu.

By agreeing to participate in the study, you acknowledge that your participation is voluntary, you are 18 years of age, and that you are aware that you may choose to terminate your participation in the study at any time and for any reason.

- ○ I consent, begin the study
- ○ I do not consent, I do not wish to participate

Date: _____

# APPENDIX E

## Recruitment Email

Hello,

You are receiving this email because we are seeking participants in a research study titled **"Usability study of a new text-based simulation modeling approach"**. As the title implies, this study focuses on assessing the usability of a new simulation modeling approach in the context of manufacturing assembly systems. An interested participant will be expected to complete the following tasks:

1. A modeling exercise using traditional simulation software (i.e., Arena),
2. A modeling exercise with a new simulation software called Automated Simulation Analysis Engine (ASAE) software.
3. A user experience questionnaire relative to the modeling exercises completed in steps 1 and 2.

Completing the above tasks will take approximately 60 minutes. A basic understanding of simulation concepts and the Arena simulation software is desired. Participation in this study will be compensated with $10.00 in cash.

This research is led by principal investigator Dr. J. David Porter and student researcher Benjamin Fields. For further information or questions about this study, please contact Dr. Porter by calling (541) 737-2446 or by email at david.porter@oregonstate.edu.

If you would like to participate in this research study, please connect via email to setup an individual study session. Thank you,

Dr. J. David Porter
david.porter@oregonstate.edu

Benjamin Fields
fieldsbe@oregonstate.edu

# APPENDIX F

## Recruitment Flyer

# Usability Study of a New Text-based Simulation Modeling Approach

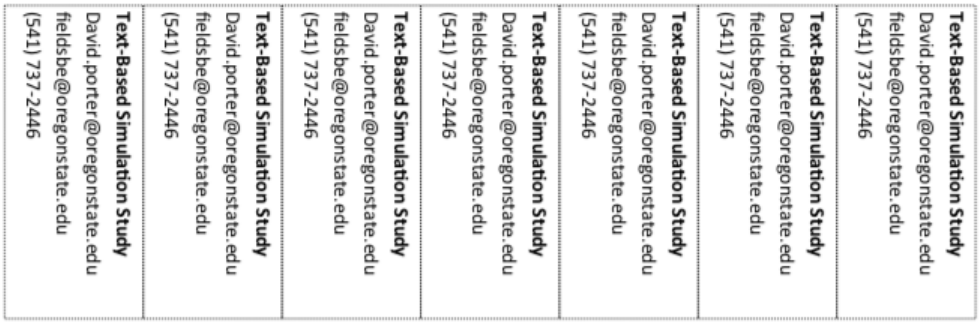
We are seeking volunteers to participate in a study that focuses on assessing the usability of a new simulation modeling approach in the context of manufacturing assembly systems. An interested participant will be expected to complete the following tasks:

1. A modeling exercise using traditional simulation software (i.e., Arena),
2. A modeling exercise with a new simulation software called Automated Simulation Analysis Engine (ASAE) software.
3. A user experience questionnaire relative to the modeling exercises completed in steps 1 and 2.

Completing the above tasks will take approximately 60 minutes. A basic understanding of simulation concepts and the Arena simulation software is desired. Participation in this study will be compensated with $10.00 in cash.

This research is led by principal investigator Dr. J. David Porter and student researcher Benjamin Fields. For further information or questions about this study, contact Dr. Porter by calling (541) 737-2446 or by email at

david.porter@oregonstate.edu, or Benjamin Fields at fieldsbe@oregonstate.edu



# APPENDIX G

## IRB Approval Notice

Oregon State University
Research Office

Human Research Protection Program
& Institutional Review Board
B308 Kerr Administration Bldg, Corvallis OR 97331
(541) 737-8008
IRB@oregonstate.edu
http://research.oregonstate.edu/irb

| | | | |
|---|---|---|---|
| Date of Notification | 07/20/2018 | | |
| Notification Type | Approval Notice | | |
| Submission Type | Initial Application | Study Number | 8715 |
| Principal Investigator | David Porter | | |
| Study Team Members | Benjamin Fields | | |
| Study Title | Usability study of a new text-based simulation modeling approach | | |
| Review Level | FLEX | | |
| Waiver(s) | Documentation of Informed Consent | | |
| Risk Level for Adults | Minimal Risk | | |
| Risk Level for Children | Study does not involve children | | |
| Funding Source | None | Cayuse Number | N/A |

**APPROVAL DATE:** 07/20/2018 **EXPIRATION DATE:** 07/19/2023
A new application will be required in order to extend the study beyond this expiration date.

**Comments**:

The above referenced study was reviewed and approved by the OSU Institutional Review Board (IRB). The IRB has determined that the protocol meets the minimum criteria for approval under the applicable regulations, state laws, and local policies.

This proposal has not been evaluated for scientific merit, except to weigh the risk to the human subjects in relation to potential benefits.

**Adding any of the following elements will invalidate the FLEX determination and require the submission of a project revision:**

- Increase in risk
- Federal funding or a plan for future federal sponsorship (e.g., proof of concept studies for federal RFPs, pilot studies intended to support a federal grant application, training and program project grants, no-cost extensions)
- Research funded or otherwise regulated by a federal agency that has signed on to the Common Rule, including all agencies within the Department of Health and Human Services
- FDA-regulated research
- NIH-issued or pending Certificate of Confidentiality
- Prisoners or parolees as subjects
- Contractual obligations or restrictions that require the application of the Common Rule or which require annual review by an IRB
- Classified research
- Clinical interventions

# APPENDIX H

## Study Protocol

# Usability study of a new text-based simulation modeling approach

I. Introduction:
   a. Introduce model in Arena
   b. Introduce model in ASAE
   c. Study Description

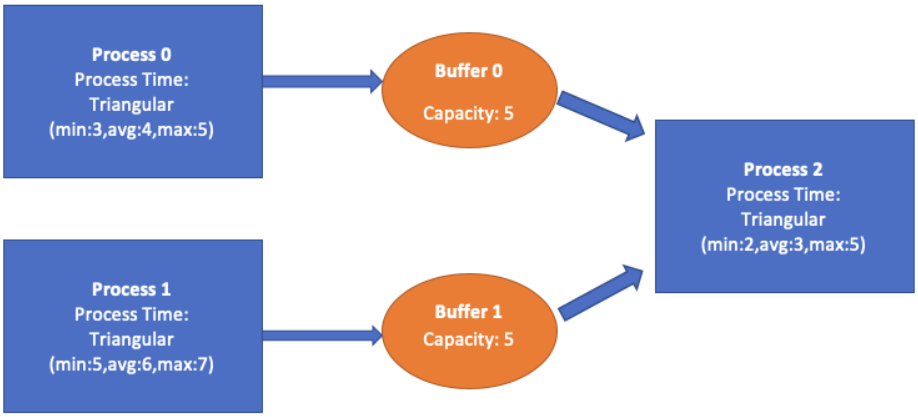
II. Module 1:
Description: The user will construct a simulation of the assembly system depicted below with the Arena simulation software package.

III. Module 2:
Description: The user will construct a simulation of the assembly system depicted below with the Automated Simulation Analysis Engine (ASAE) simulation software.

IV. Module 3:
Description: Complete simulation modeling user experience questionnaire

**Assembly System to be Modeled**

Number of Jobs: 100




**Expected Simulation Runtime is approximately 603 time units**

**Module 1: Arena**

Model the Assembly System using the following steps in Arena

1. Define resources
   - Worker 0: 1
   - Worker 1: 1
   - Worker 2: 1
   - Buffer 0: 5
   - Buffer 1: 5

2. Drag in Create modules
   - One for Process 0 branch
   - One for Process 1 branch

3. Define Create modules
   - Entities per arrival 1
   - For process 0 branch time between arrivals Expo 1 minute
   - For process 1 branch time between arrivals Expo 2 minutes
   - Time units minutes
   - Max Arrivals 100
   - First arrival at time 0

4. Drag in Seize modules for Processes 0 and 1
   - Process 0 – Seize resource worker 0
   - Process 1 – seize resource worker 1

5. Insert Delay module for Processes 0 and 1

6. Define the time for Processes 0 and 1
   - Time units minutes

7. Drag in Seize module for buffers
   - Seize resource buffer 0
   - Seize resource buffer 1

8. Drag in Release modules for Processes 0 and 1
   - Release resource worker 0
   - Release resource worker 1

9. Drag in Match module and connect Process 0 branch and Process 1 branch

10. Drag in Batch module and set size to 2

11. Drag in Seize module for Process 2
    - Seize resource worker 2

12. Drag in Release module
    - Release one of resource buffer 0
    - Release one of resource buffer 1

13. Drag in and define Delay module for Process 2

14. Drag in Release module for Process 2
    - Release resource worker 2

15. Drag in Dispose module

16. Open Run parameters

17. Set all time units to minutes

18. Run

19. Review results to gain insight into process

**Module 2: Automated Simulation Analysis Engine**

1. Model the Assembly System using the following steps
2. Navigate to the site found at this link http://35.196.62.92/
3. Enter the number of jobs to simulate : 100
4. Enter the number of Processes: 3
5. Press Define
6. Define the Characteristics of each process in the Process Panel
7. Define Process 0
    a. Process Time
        i. Note the format of a triangular Distribution Process Time T:low:avg:max
        ii. Triangular with min:3 avg:4 and max:5
        **iii. Enter as T:3:4:5**
    b. Position Type
        **i. Enter 0**
    c. Downstream Connections
        i. Used to indicate the number of possible paths a job can go
        ii. Format is number,PID(percentage)buffer_capacity,... (X,XX(X.XX)XX,...)
        **iii. Enter as 1,02(1.00)05**
    d. Upstream Connections
        i. No upstream connections
        **ii. Enter 0**
8. Press ADD
9. Define Process 1
    a. Process Time
        i. Note the format of a triangular Distribution Process Time T:low:avg:max
        ii. Triangular with min:5 avg:6 and max:7
        **iii. Enter as T:5:6:7**
    b. Position Type
        **i. Enter 0**
    c. Downstream Connections
        i. Used to indicate the number of possible paths a job can go

ii. Format is number,PID(percentage)buffer_capacity,... (X,XX(X.XX)XX,...)
**iii. Enter as 1,02(1.00)05**

d. Upstream Connections
i. No upstream connections
**ii. Enter 0**

10. Press ADD

11. Define Process 2
a. Process Time
i. Note the format of a triangular Distribution Process Time T:low:avg:max
ii. Triangular with min:2 avg:3 and max:5
**iii. Enter as T:2:3:5**

b. Position Type
**i. Enter 2**

c. Downstream Connections
i. No downstream Connections
**ii. Enter 0**

d. Upstream Connections
i. Used to indicate the number of incoming paths
ii. Format is Number,(PID,Buffer_Index),... X,(XX,X),...

**iii. Enter as 2,(00,0),(01,0)**

12. Press ADD

13. Press DOWNLOAD

14. Provide the name you would like to use

15. Execute ASAE with Model file
a. Cmd to type in terminal **./ASAE path/to/your/model/file.txt**
b. Drag the model file into the terminal

16. Review results and think about the characteristics of the system

**Module 3**

Please complete the simulation modeling user experience questionnaire available via the following link:

http://oregonstate.qualtrics.com/jfe/form/SV_8ctZ1jYTexVkPcN

# APPENDIX I

## Questionaire

# Simulation Modeling Questionnaire

Thank you for participating in this study and taking the time to complete this questionnaire.

**Purpose:** The purpose of this questionnaire is to reflect on the previously completed modeling exercises and to provide feedback based on your experience when modeling with Arena and the Automated Simulation Analysis Engine (ASAE). The questions will focus on the interface used to model a process (i.e., text-based and Graphical User Interface (GUI) modules), the ease of use, and the information that can be obtained after completing the modeling exercise with both Arena and ASAE.

**Data Collection:** The data collected within this study will be used and published in support of a M.S. thesis, but no personally identifiable information will be collected or shared.

**Benefit:** Your well thought out responses will help to better understand how new modeling approaches can be used and how text-based simulation technologies can save engineers valuable time and resources, so please take your time and answer the questionnaire to the best of your ability.

Although it is preferable that you answer all the questions included in the questionnaire, please feel free to skip any questions you wish. Your participation in this research is voluntary and your decision to take part or not take part in this study will not affect your grades, your relationship with your professors, or your standing in the University. Finally, you have the right to withdraw at any point during the study, for any reason, and without any prejudice.

---

**Start of Block: Rate Skill block**

Q1 After going through the pre-study training exercises on Arena and Automated Simulation Analysis Engine (ASAE), I rate my ability to develop discrete event simulation (DES) models as:

- ○ Excellent (1)
- ○ Good (2)
- ○ Average (3)
- ○ Poor (4)

Q2 DES-based modeling approaches have different levels of complexity, which affect the time required to construct a model of a manufacturing assembly

system. Please select the option that best reflects your experience when constructing a model of a manufacturing assembly system with Arena and ASAE.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| Constructing a model of a manufacturing assembly system took less time with ASAE than with Arena (1) | ○ | ○ | ○ | ○ | ○ | ○ |
| ASAE reduced the time needed to construct a model of a manufacturing assembly system when compared to Arena (3) | ○ | ○ | ○ | ○ | ○ | ○ |
| ASAE has no impact on the time needed to construct a model of a manufacturing assembly system (5) | ○ | ○ | ○ | ○ | ○ | ○ |

Q3 DES-based modeling approaches simulate manufacturing assembly systems faster than others. Please select the option that best reflects your experience simulating a manufacturing assembly system with Arena and ASAE.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| Simulating a manufacturing assembly system took less time with Arena than with ASAE (2) | ○ | ○ | ○ | ○ | ○ | ○ |
| When compared to Arena, ASAE reduced the time needed to simulate a manufacturing assembly system (3) | ○ | ○ | ○ | ○ | ○ | ○ |
| When compared to Arena, ASAE has no impact on the time needed to simulate a manufacturing assembly system (5) | ○ | ○ | ○ | ○ | ○ | ○ |

Q4 Graphical user interfaces (GUI) are used within Arena to construct and simulate a manufacturing assembly system, whereas ASAE uses a text-based interface for the same purpose. Please select the option that best reflects your opinion about using a GUI and a text-based interface to construct and simulate a manufacturing assembly system.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| Using Arena's GUI to construct and simulate a manufacturing assembly system is easier than using ASAE's text-based interface (1) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using ASAE's text-based interface to construct and simulate a manufacturing assembly system is easier than using Arena's GUI (2) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using ASAE's text-based interface to construct and simulate a manufacturing assembly system is more difficult than using Arena's GUI (3) | ○ | ○ | ○ | ○ | ○ | ○ |

| | | | | | | |
|---|---|---|---|---|---|---|
| I would rather construct and simulate a manufacturing assembly system using Arena's GUI instead of ASAE's text-based interface (4) | ○ | ○ | ○ | ○ | ○ | ○ |
| I would rather construct and simulate a manufacturing assembly system using ASAE's text-based interface instead of Arena's GUI (5) | ○ | ○ | ○ | ○ | ○ | ○ |
| Both Arena's GUI and ASAE's text-based interface are equivalent when constructing and simulating a manufacturing assembly system (6) | ○ | ○ | ○ | ○ | ○ | ○ |

**End of Block: Modeling Block**

**Start of Block: Block 6**

Q4 Two common features of manufacturing assembly systems are parallel workstations and finite capacity buffers. Please rate your experience when modeling parallel workstations and finite capacity buffers with Arena and ASAE.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| Modeling parallel workstations with ASAE was difficult (1) | ○ | ○ | ○ | ○ | ○ | ○ |
| Modeling parallel workstations with Arena was easy (2) | ○ | ○ | ○ | ○ | ○ | ○ |
| Compared to Arena, it was more difficult to model parallel workstations with ASAE (3) | ○ | ○ | ○ | ○ | ○ | ○ |
| Modeling finite capacity buffers with ASAE was difficult (4) | ○ | ○ | ○ | ○ | ○ | ○ |
| Modeling finite capacity buffers with Arena was easy (5) | ○ | ○ | ○ | ○ | ○ | ○ |

| | | | | | | |
|---|---|---|---|---|---|---|
| Compared to Arena, it was more difficult to model finite capacity buffers with ASAE (6) | ○ | ○ | ○ | ○ | ○ | ○ |

Q7 Arena uses a GUI and ASAE uses a text-based interface. Please select the option that best reflects your opinion about how well you understood the

characteristics of a manufacturing assembly system when modeling it with a GUI as opposed to a text-based interface.as opposed to a text-based interface.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| Using Arena's GUI modules allowed me to understand process flow (1) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using ASAE's text-based approach allowed me to understand process flow (2) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using Arena's GUI modules allowed me to understand the characteristics of each process (3) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using ASAE's text-based approach allowed me to understand the characteristics of each process (4) | ○ | ○ | ○ | ○ | ○ | ○ |

| | | | | | | |
|---|---|---|---|---|---|---|
| Using Arena's GUI modules allowed me to get a better initial understanding of the system (5) | ○ | ○ | ○ | ○ | ○ | ○ |
| Using ASAE's text-based approach allowed me to get a better initial understanding of the system (6) | ○ | ○ | ○ | ○ | ○ | ○ |

Q17 DES-based modeling approaches generate useful results that help in gaining a better understanding of the performance of a manufacturing assembly

system. Please select the option that reflects your opinion about the value of the results generated by Arena and ASAE.

| | Strongly agree (1) | Agree (2) | Somewhat agree (3) | Somewhat disagree (4) | Disagree (5) | Strongly disagree (6) |
|---|---|---|---|---|---|---|
| I was able to understand process flow using the Arena results. (1) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to understand process flow using the ASAE results. (2) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to identify potential bottlenecks using the Arena results. (3) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to identify potential bottlenecks using the ASAE results. (4) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to understand the throughput of the system with the Arena results (5) | ○ | ○ | ○ | ○ | ○ | ○ |

| | | | | | | |
|---|---|---|---|---|---|---|
| I was able to understand the throughput of the system with the ASAE results (6) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to understand process times with the results provided by Arena (7) | ○ | ○ | ○ | ○ | ○ | ○ |
| I was able to understand process times with the results provided by ASAE (8) | ○ | ○ | ○ | ○ | ○ | ○ |
| The results provided by Arena allow me to better understand the manufacturing assembly system (9) | ○ | ○ | ○ | ○ | ○ | ○ |
| The results provided by ASAE allow me to better understand the manufacturing assembly system (10) | ○ | ○ | ○ | ○ | ○ | ○ |

End of Block: Block 7

Start of Block: Compare Block

Q10 For each statement in this section, please select how the modeling approach enabled by either Arena or ASAE relates to the statement. The closer a number is to a specific modeling approach, the stronger the connection is between that statement and the modeling approach.

Q9 Saves more time in creating and simulating a manufacturing assembly system.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q11 Increases the time needed to create and simulate a manufacturing assembly system.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q12 Facilitates the modeling of finite capacity buffers.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q13 Facilitates the modeling of parallel workstations.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q14 The modeling approach is easy to use.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q15 The modeling approach is difficult to understand.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q17 The data generated by the modeling approach provides more insight into the performance of the manufacturing assembly system.

| | 1 (1) | 2 (2) | 3 (3) | 4 (4) | 5 (5) | 6 (6) | |
|---|---|---|---|---|---|---|---|
| ASAE | | | | | | | Arena |

Q8 Based on your user experience, which modeling approach would you rather use to create and simulate a manufacturing assembly system?

- ASAE (1)
- Arena (2)

# APPENDIX J

## Example of Start.txt file created by ASAE simulation engine

JobID Start,StartTime,Resource,JobsInSystem
[1:0-(x)],0.000000,1
[1:1-(x)],0.000000,2
[2:0-(x)],3.616230,3
[2:1-(x)],5.518668,4
[1:2-([1:0-(x)][1:1-(x)])],5.518783,4
[3:0-(x)],7.817717,5
[4:0-(x)],11.462727,4
[3:1-(x)],11.797397,5
[2:2-([2:0-(x)][2:1-(x)])],11.797658,5
[5:0-(x)],15.045105,6
[4:1-(x)],17.384474,5
[3:2-([3:0-(x)][3:1-(x)])],17.385027,5
[6:0-(x)],19.402149,6
[5:1-(x)],22.703205,5
[4:2-([4:0-(x)][4:1-(x)])],22.704079,5
[7:0-(x)],22.868164,6
[8:0-(x)],26.514030,7
[6:1-(x)],29.526850,6
[5:2-([5:0-(x)][5:1-(x)])],29.526886,6
[9:0-(x)],30.473555,7
[10:0-(x)],35.161819,6
[7:1-(x)],36.437546,7
[6:2-([6:0-(x)][6:1-(x)])],36.438168,7
[11:0-(x)],39.017387,8
[8:1-(x)],42.269699,7
[7:2-([7:0-(x)][7:1-(x)])],42.270237,7
[12:0-(x)],42.413597,8
[13:0-(x)],46.532181,7
[9:1-(x)],47.898106,8
[8:2-([8:0-(x)][8:1-(x)])],47.899040,8
[14:0-(x)],50.387115,9
[10:1-(x)],53.603619,8
[9:2-([9:0-(x)][9:1-(x)])],53.604435,8
[15:0-(x)],54.031441,9
[11:1-(x)],59.886330,8
[10:2-([10:0-(x)][10:1-(x)])],59.887295,8
[16:0-(x)],59.887985,9

[12:1-(x)],66.180092,8
[11:2-([11:0-(x)][11:1-(x)])],66.180611,8
[17:0-(x)],66.181084,9
[13:1-(x)],71.670044,8
[12:2-([12:0-(x)][12:1-(x)])],71.670731,8
[18:0-(x)],71.671135,9
[14:1-(x)],77.547501,8
[13:2-([13:0-(x)][13:1-(x)])],77.548218,8
[19:0-(x)],77.548790,9
[15:1-(x)],83.454872,8
[14:2-([14:0-(x)][14:1-(x)])],83.455254,8
[20:0-(x)],83.455841,9
[16:1-(x)],89.288864,8
[15:2-([15:0-(x)][15:1-(x)])],89.288986,8
[21:0-(x)],89.289879,9
[17:1-(x)],95.878166,8
[16:2-([16:0-(x)][16:1-(x)])],95.878845,8
[22:0-(x)],95.878860,9
[18:1-(x)],101.772667,8
[17:2-([17:0-(x)][17:1-(x)])],101.773399,8
[23:0-(x)],101.773659,9
[19:1-(x)],107.295792,8
[18:2-([18:0-(x)][18:1-(x)])],107.296646,8
[24:0-(x)],107.296967,9
[20:1-(x)],113.034775,8
[19:2-([19:0-(x)][19:1-(x)])],113.034805,8
[25:0-(x)],113.035454,9
[21:1-(x)],118.967079,8
[20:2-([20:0-(x)][20:1-(x)])],118.967323,8
[26:0-(x)],118.967422,9
[22:1-(x)],125.450386,8
[21:2-([21:0-(x)][21:1-(x)])],125.450531,8
[27:0-(x)],125.451073,9
[23:1-(x)],131.410324,8
[22:2-([22:0-(x)][22:1-(x)])],131.410919,8
[28:0-(x)],131.411240,9
[24:1-(x)],137.803421,8
[23:2-([23:0-(x)][23:1-(x)])],137.804352,8
[29:0-(x)],137.804413,9
[25:1-(x)],143.574615,8
[24:2-([24:0-(x)][24:1-(x)])],143.575272,8
[30:0-(x)],143.575378,9
[26:1-(x)],148.808990,8

[25:2-([25:0-(x)][25:1-(x)])],148.809891,8
[31:0-(x)],148.809921,9
[27:1-(x)],153.981659,8
[26:2-([26:0-(x)][26:1-(x)])],153.982330,8
[32:0-(x)],153.982407,9
[28:1-(x)],160.270020,8
[27:2-([27:0-(x)][27:1-(x)])],160.270859,8
[33:0-(x)],160.271454,9
[29:1-(x)],165.996857,8
[28:2-([28:0-(x)][28:1-(x)])],165.997818,8
[34:0-(x)],165.998520,9
[30:1-(x)],172.113586,8
[29:2-([29:0-(x)][29:1-(x)])],172.114410,8
[35:0-(x)],172.114502,9
[31:1-(x)],178.396713,8
[30:2-([30:0-(x)][30:1-(x)])],178.397324,8
[36:0-(x)],178.397949,9
[32:1-(x)],184.407379,8
[31:2-([31:0-(x)][31:1-(x)])],184.407425,8
[37:0-(x)],184.407761,9
[33:1-(x)],189.877884,8
[32:2-([32:0-(x)][32:1-(x)])],189.878815,8
[38:0-(x)],189.879791,9
[34:1-(x)],196.268860,8
[33:2-([33:0-(x)][33:1-(x)])],196.268906,8
[39:0-(x)],196.269165,9
[35:1-(x)],201.989410,8
[34:2-([34:0-(x)][34:1-(x)])],201.989456,8
[40:0-(x)],201.989990,9
[36:1-(x)],208.088531,8
[35:2-([35:0-(x)][35:1-(x)])],208.088699,8
[41:0-(x)],208.089203,9
[37:1-(x)],213.932007,8
[36:2-([36:0-(x)][36:1-(x)])],213.932037,8
[42:0-(x)],213.932709,9
[38:1-(x)],219.473267,8
[37:2-([37:0-(x)][37:1-(x)])],219.474167,8
[43:0-(x)],219.475006,9
[39:1-(x)],225.918808,8
[38:2-([38:0-(x)][38:1-(x)])],225.919601,8
[44:0-(x)],225.920013,9
[40:1-(x)],232.152252,8
[39:2-([39:0-(x)][39:1-(x)])],232.152893,8

[45:0-(x)],232.152985,9
[41:1-(x)],237.979507,8
[40:2-([40:0-(x)][40:1-(x)])],237.979568,8
[46:0-(x)],237.979828,9
[42:1-(x)],244.608871,8
[41:2-([41:0-(x)][41:1-(x)])],244.608948,8
[47:0-(x)],244.609573,9
[43:1-(x)],251.152237,8
[42:2-([42:0-(x)][42:1-(x)])],251.152939,8
[48:0-(x)],251.153534,9
[44:1-(x)],256.370148,8
[43:2-([43:0-(x)][43:1-(x)])],256.370300,8
[49:0-(x)],256.371063,9
[45:1-(x)],262.818665,8
[44:2-([44:0-(x)][44:1-(x)])],262.819397,8
[50:0-(x)],262.820099,9
[46:1-(x)],268.675323,8
[45:2-([45:0-(x)][45:1-(x)])],268.675629,8
[51:0-(x)],268.675964,9
[47:1-(x)],273.760620,8
[46:2-([46:0-(x)][46:1-(x)])],273.760651,8
[52:0-(x)],273.761108,9
[48:1-(x)],280.304199,8
[47:2-([47:0-(x)][47:1-(x)])],280.304260,8
[53:0-(x)],280.305206,9
[49:1-(x)],286.249146,8
[48:2-([48:0-(x)][48:1-(x)])],286.250000,8
[54:0-(x)],286.250549,9
[50:1-(x)],292.696228,8
[49:2-([49:0-(x)][49:1-(x)])],292.696960,8
[55:0-(x)],292.697815,9
[51:1-(x)],298.745880,8
[50:2-([50:0-(x)][50:1-(x)])],298.746704,8
[56:0-(x)],298.746796,9
[52:1-(x)],305.018738,8
[51:2-([51:0-(x)][51:1-(x)])],305.019501,8
[57:0-(x)],305.020294,9
[53:1-(x)],311.880951,8
[52:2-([52:0-(x)][52:1-(x)])],311.881042,8
[58:0-(x)],311.881805,9
[54:1-(x)],318.546417,8
[53:2-([53:0-(x)][53:1-(x)])],318.546631,8
[59:0-(x)],318.547363,9

[55:1-(x)],324.075165,8
[54:2-([54:0-(x)][54:1-(x)])],324.075470,8
[60:0-(x)],324.075500,9
[56:1-(x)],330.809875,8
[55:2-([55:0-(x)][55:1-(x)])],330.810150,8
[61:0-(x)],330.810272,9
[57:1-(x)],337.314575,8
[56:2-([56:0-(x)][56:1-(x)])],337.314911,8
[62:0-(x)],337.315430,9
[58:1-(x)],343.223083,8
[57:2-([57:0-(x)][57:1-(x)])],343.224060,8
[63:0-(x)],343.224365,9
[59:1-(x)],348.941925,8
[58:2-([58:0-(x)][58:1-(x)])],348.942841,8
[64:0-(x)],348.943176,9
[60:1-(x)],355.122070,8
[59:2-([59:0-(x)][59:1-(x)])],355.122253,8
[65:0-(x)],355.122314,9
[61:1-(x)],361.706787,8
[60:2-([60:0-(x)][60:1-(x)])],361.707031,8
[66:0-(x)],361.707123,9
[62:1-(x)],367.720947,8
[61:2-([61:0-(x)][61:1-(x)])],367.721039,8
[67:0-(x)],367.721649,9
[63:1-(x)],373.870850,8
[62:2-([62:0-(x)][62:1-(x)])],373.871063,8
[68:0-(x)],373.871429,9
[64:1-(x)],379.427246,8
[63:2-([63:0-(x)][63:1-(x)])],379.427734,8
[69:0-(x)],379.428375,9
[65:1-(x)],385.117981,8
[64:2-([64:0-(x)][64:1-(x)])],385.118134,8
[70:0-(x)],385.119049,9
[66:1-(x)],390.780609,8
[65:2-([65:0-(x)][65:1-(x)])],390.781189,8
[71:0-(x)],390.781525,9
[67:1-(x)],397.339203,8
[66:2-([66:0-(x)][66:1-(x)])],397.339294,8
[72:0-(x)],397.339752,9
[68:1-(x)],403.421600,8
[67:2-([67:0-(x)][67:1-(x)])],403.422058,8
[73:0-(x)],403.422821,9
[69:1-(x)],408.902985,8

[68:2-([68:0-(x)][68:1-(x)])],408.903137,8
[74:0-(x)],408.903259,9
[70:1-(x)],414.924500,8
[69:2-([69:0-(x)][69:1-(x)])],414.925049,8
[75:0-(x)],414.925415,9
[71:1-(x)],421.118256,8
[70:2-([70:0-(x)][70:1-(x)])],421.118530,8
[76:0-(x)],421.119232,9
[72:1-(x)],427.101166,8
[71:2-([71:0-(x)][71:1-(x)])],427.101379,8
[77:0-(x)],427.101746,9
[73:1-(x)],433.107513,8
[72:2-([72:0-(x)][72:1-(x)])],433.107788,8
[78:0-(x)],433.107880,9
[74:1-(x)],439.568542,8
[73:2-([73:0-(x)][73:1-(x)])],439.568939,8
[79:0-(x)],439.569916,9
[75:1-(x)],445.540558,8
[74:2-([74:0-(x)][74:1-(x)])],445.540802,8
[80:0-(x)],445.541016,9
[76:1-(x)],451.721344,8
[75:2-([75:0-(x)][75:1-(x)])],451.721588,8
[81:0-(x)],451.721802,9
[77:1-(x)],457.781708,8
[76:2-([76:0-(x)][76:1-(x)])],457.781738,8
[82:0-(x)],457.782379,9
[78:1-(x)],464.545929,8
[77:2-([77:0-(x)][77:1-(x)])],464.546600,8
[83:0-(x)],464.547546,9
[79:1-(x)],470.863220,8
[78:2-([78:0-(x)][78:1-(x)])],470.864166,8
[84:0-(x)],470.864777,9
[80:1-(x)],476.427917,8
[79:2-([79:0-(x)][79:1-(x)])],476.428864,8
[85:0-(x)],476.429108,9
[81:1-(x)],482.226379,8
[80:2-([80:0-(x)][80:1-(x)])],482.227234,8
[86:0-(x)],482.227600,9
[82:1-(x)],488.489410,8
[81:2-([81:0-(x)][81:1-(x)])],488.489563,8
[87:0-(x)],488.489868,9
[83:1-(x)],494.277527,8
[82:2-([82:0-(x)][82:1-(x)])],494.277771,8

[88:0-(x)],494.277832,9
[84:1-(x)],499.980133,8
[83:2-([83:0-(x)][83:1-(x)])],499.980804,8
[89:0-(x)],499.981598,9
[85:1-(x)],506.960724,8
[84:2-([84:0-(x)][84:1-(x)])],506.961548,8
[90:0-(x)],506.962036,9
[86:1-(x)],512.848206,8
[85:2-([85:0-(x)][85:1-(x)])],512.848999,8
[91:0-(x)],512.849487,9
[87:1-(x)],519.240662,8
[86:2-([86:0-(x)][86:1-(x)])],519.241638,8
[92:0-(x)],519.241821,9
[88:1-(x)],524.380554,8
[87:2-([87:0-(x)][87:1-(x)])],524.381042,8
[93:0-(x)],524.381165,9
[89:1-(x)],529.832153,8
[88:2-([88:0-(x)][88:1-(x)])],529.832336,8
[94:0-(x)],529.833008,9
[90:1-(x)],535.463013,8
[89:2-([89:0-(x)][89:1-(x)])],535.463379,8
[95:0-(x)],535.463440,9
[91:1-(x)],541.695801,8
[90:2-([90:0-(x)][90:1-(x)])],541.696716,8
[96:0-(x)],541.696777,9
[92:1-(x)],547.317627,8
[91:2-([91:0-(x)][91:1-(x)])],547.318604,8
[97:0-(x)],547.318726,9
[93:1-(x)],553.848022,8
[92:2-([92:0-(x)][92:1-(x)])],553.848511,8
[98:0-(x)],553.848633,9
[94:1-(x)],560.126892,8
[93:2-([93:0-(x)][93:1-(x)])],560.127258,8
[99:0-(x)],560.127319,9
[95:1-(x)],566.660889,8
[94:2-([94:0-(x)][94:1-(x)])],566.661316,8
[100:0-(x)],566.662109,9
[96:1-(x)],573.005127,8
[95:2-([95:0-(x)][95:1-(x)])],573.006042,8
[101:0-(x)],573.006470,9
[97:1-(x)],579.006714,8
[96:2-([96:0-(x)][96:1-(x)])],579.007019,8
[102:0-(x)],579.007874,9

[98:1-(x)],585.816223,8
[97:2-([97:0-(x)][97:1-(x)])],585.816345,8
[103:0-(x)],585.816895,9
[99:1-(x)],591.849487,8
[98:2-([98:0-(x)][98:1-(x)])],591.850098,8
[104:0-(x)],591.850708,9
[100:1-(x)],597.328430,8
[99:2-([99:0-(x)][99:1-(x)])],597.329285,8
[105:0-(x)],597.330017,9
[101:1-(x)],603.438416,8
[100:2-([100:0-(x)][100:1-(x)])],603.438477,8
[106:0-(x)],603.438904,9

# APPENDIX K

## Example of Finish.txt file created by ASAE simulation engine

JobID END,ExitTime,JobsInSystem
[1:0-(x)],3.616230,2
[1:1-(x)],5.518668,3
[2:0-(x)],7.817717,4
[1:2-([1:0-(x)][1:1-(x)])],8.338260,3
[3:0-(x)],11.462727,3
[2:1-(x)],11.797397,4
[4:0-(x)],15.045105,5
[2:2-([2:0-(x)][2:1-(x)])],15.393543,4
[3:1-(x)],17.384474,4
[5:0-(x)],19.402149,5
[3:2-([3:0-(x)][3:1-(x)])],21.446770,4
[4:1-(x)],22.703205,4
[6:0-(x)],22.868164,5
[7:0-(x)],26.514030,6
[4:2-([4:0-(x)][4:1-(x)])],26.666458,5
[5:1-(x)],29.526850,5
[8:0-(x)],30.473555,6
[5:2-([5:0-(x)][5:1-(x)])],34.263363,5
[9:0-(x)],35.161819,5
[6:1-(x)],36.437546,6
[10:0-(x)],39.017387,7
[6:2-([6:0-(x)][6:1-(x)])],39.291874,6
[7:1-(x)],42.269699,6
[11:0-(x)],42.413597,7
[7:2-([7:0-(x)][7:1-(x)])],46.036064,6
[12:0-(x)],46.532181,6
[8:1-(x)],47.898106,7
[13:0-(x)],50.387115,8
[8:2-([8:0-(x)][8:1-(x)])],52.227192,7
[9:1-(x)],53.603619,7
[14:0-(x)],54.031441,8
[9:2-([9:0-(x)][9:1-(x)])],57.249744,7
[15:0-(x)],58.221901,7
[10:1-(x)],59.886330,7
[10:2-([10:0-(x)][10:1-(x)])],63.765938,7
[16:0-(x)],63.767406,7
[11:1-(x)],66.180092,7

[11:2-([11:0-(x)][11:1-(x)])],69.147118,7
[17:0-(x)],69.977066,7
[12:1-(x)],71.670044,7
[12:2-([12:0-(x)][12:1-(x)])],74.371964,7
[18:0-(x)],76.087593,7
[13:1-(x)],77.547501,7
[13:2-([13:0-(x)][13:1-(x)])],81.677231,7
[19:0-(x)],82.224518,7
[14:1-(x)],83.454872,7
[14:2-([14:0-(x)][14:1-(x)])],85.778915,7
[20:0-(x)],87.975601,7
[15:1-(x)],89.288864,7
[21:0-(x)],93.059410,9
[15:2-([15:0-(x)][15:1-(x)])],93.128357,7
[16:1-(x)],95.878166,7
[22:0-(x)],99.403961,9
[16:2-([16:0-(x)][16:1-(x)])],100.239243,7
[17:1-(x)],101.772667,7
[23:0-(x)],105.554924,9
[17:2-([17:0-(x)][17:1-(x)])],105.792473,7
[18:1-(x)],107.295792,7
[18:2-([18:0-(x)][18:1-(x)])],109.991478,7
[24:0-(x)],111.613235,7
[19:1-(x)],113.034775,7
[19:2-([19:0-(x)][19:1-(x)])],115.967972,7
[25:0-(x)],117.079460,7
[20:1-(x)],118.967079,7
[20:2-([20:0-(x)][20:1-(x)])],122.719032,7
[26:0-(x)],122.735565,7
[21:1-(x)],125.450386,7
[21:2-([21:0-(x)][21:1-(x)])],129.472290,7
[27:0-(x)],129.851273,7
[22:1-(x)],131.410324,7
[22:2-([22:0-(x)][22:1-(x)])],135.631073,7
[28:0-(x)],136.067200,7
[23:1-(x)],137.803421,7
[23:2-([23:0-(x)][23:1-(x)])],142.245926,7
[29:0-(x)],142.771729,7
[24:1-(x)],143.574615,7
[30:0-(x)],146.836044,9
[24:2-([24:0-(x)][24:1-(x)])],147.527878,7
[25:1-(x)],148.808990,7
[25:2-([25:0-(x)][25:1-(x)])],152.626801,7

[31:0-(x)],152.768875,7
[26:1-(x)],153.981659,7
[26:2-([26:0-(x)][26:1-(x)])],156.802475,7
[32:0-(x)],157.775909,7
[27:1-(x)],160.270020,7
[27:2-([27:0-(x)][27:1-(x)])],163.959488,7
[33:0-(x)],164.399277,7
[28:1-(x)],165.996857,7
[28:2-([28:0-(x)][28:1-(x)])],170.301666,7
[34:0-(x)],170.722717,7
[29:1-(x)],172.113586,7
[29:2-([29:0-(x)][29:1-(x)])],175.841354,7
[35:0-(x)],176.033325,7
[30:1-(x)],178.396713,7
[36:0-(x)],182.528549,9
[30:2-([30:0-(x)][30:1-(x)])],182.591660,7
[31:1-(x)],184.407379,7
[31:2-([31:0-(x)][31:1-(x)])],187.382263,7
[37:0-(x)],187.556458,7
[32:1-(x)],189.877884,7
[32:2-([32:0-(x)][32:1-(x)])],192.178146,7
[38:0-(x)],194.542023,7
[33:1-(x)],196.268860,7
[39:0-(x)],200.553894,9
[33:2-([33:0-(x)][33:1-(x)])],200.894760,7
[34:1-(x)],201.989410,7
[40:0-(x)],205.226074,9
[34:2-([34:0-(x)][34:1-(x)])],206.253799,7
[35:1-(x)],208.088531,7
[41:0-(x)],211.721161,9
[35:2-([35:0-(x)][35:1-(x)])],212.078003,7
[36:1-(x)],213.932007,7
[42:0-(x)],217.533356,9
[36:2-([36:0-(x)][36:1-(x)])],218.563766,7
[37:1-(x)],219.473267,7
[43:0-(x)],223.118317,9
[37:2-([37:0-(x)][37:1-(x)])],223.269974,7
[38:1-(x)],225.918808,7
[44:0-(x)],229.522491,9
[38:2-([38:0-(x)][38:1-(x)])],230.557678,7
[39:1-(x)],232.152252,7
[39:2-([39:0-(x)][39:1-(x)])],234.317825,7
[45:0-(x)],236.044220,7

[40:1-(x)],237.979507,7
[40:2-([40:0-(x)][40:1-(x)])],241.857605,7
[46:0-(x)],242.393997,7
[41:1-(x)],244.608871,7
[47:0-(x)],248.419312,9
[41:2-([41:0-(x)][41:1-(x)])],248.722855,7
[42:1-(x)],251.152237,7
[48:0-(x)],255.296524,9
[42:2-([42:0-(x)][42:1-(x)])],255.952377,7
[43:1-(x)],256.370148,7
[43:2-([43:0-(x)][43:1-(x)])],259.288574,7
[49:0-(x)],259.788788,7
[44:1-(x)],262.818665,7
[44:2-([44:0-(x)][44:1-(x)])],265.653381,7
[50:0-(x)],266.919617,7
[45:1-(x)],268.675323,7
[45:2-([45:0-(x)][45:1-(x)])],271.317474,7
[51:0-(x)],273.284760,7
[46:1-(x)],273.760620,7
[46:2-([46:0-(x)][46:1-(x)])],277.378693,7
[52:0-(x)],277.893250,7
[47:1-(x)],280.304199,7
[53:0-(x)],284.431763,9
[47:2-([47:0-(x)][47:1-(x)])],284.637665,7
[48:1-(x)],286.249146,7
[54:0-(x)],290.390594,9
[48:2-([48:0-(x)][48:1-(x)])],290.723083,7
[49:1-(x)],292.696228,7
[55:0-(x)],296.428497,9
[49:2-([49:0-(x)][49:1-(x)])],296.439484,7
[50:1-(x)],298.745880,7
[50:2-([50:0-(x)][50:1-(x)])],302.911682,7
[56:0-(x)],302.951752,7
[51:1-(x)],305.018738,7
[57:0-(x)],308.485931,9
[51:2-([51:0-(x)][51:1-(x)])],308.782257,7
[52:1-(x)],311.880951,7
[58:0-(x)],315.414337,9
[52:2-([52:0-(x)][52:1-(x)])],315.756012,7
[53:1-(x)],318.546417,7
[59:0-(x)],322.287933,9
[53:2-([53:0-(x)][53:1-(x)])],322.339264,7
[54:1-(x)],324.075165,7

[54:2-([54:0-(x)][54:1-(x)])],328.040680,7
[60:0-(x)],328.675262,7
[55:1-(x)],330.809875,7
[55:2-([55:0-(x)][55:1-(x)])],334.496094,7
[61:0-(x)],334.567108,7
[56:1-(x)],337.314575,7
[62:0-(x)],341.297821,9
[56:2-([56:0-(x)][56:1-(x)])],342.009521,7
[57:1-(x)],343.223083,7
[57:2-([57:0-(x)][57:1-(x)])],346.876312,7
[63:0-(x)],347.330841,7
[58:1-(x)],348.941925,7
[58:2-([58:0-(x)][58:1-(x)])],352.581390,7
[64:0-(x)],353.465668,7
[59:1-(x)],355.122070,7
[65:0-(x)],359.180359,9
[59:2-([59:0-(x)][59:1-(x)])],359.267883,7
[60:1-(x)],361.706787,7
[66:0-(x)],365.826324,9
[60:2-([60:0-(x)][60:1-(x)])],366.387665,7
[61:1-(x)],367.720947,7
[61:2-([61:0-(x)][61:1-(x)])],372.178162,7
[67:0-(x)],372.286285,7
[62:1-(x)],373.870850,7
[62:2-([62:0-(x)][62:1-(x)])],377.681458,7
[68:0-(x)],378.075867,7
[63:1-(x)],379.427246,7
[63:2-([63:0-(x)][63:1-(x)])],383.169434,7
[69:0-(x)],383.789703,7
[64:1-(x)],385.117981,7
[70:0-(x)],388.589111,9
[64:2-([64:0-(x)][64:1-(x)])],389.157776,7
[65:1-(x)],390.780609,7
[71:0-(x)],394.055664,9
[65:2-([65:0-(x)][65:1-(x)])],394.837708,7
[66:1-(x)],397.339203,7
[72:0-(x)],400.703705,9
[66:2-([66:0-(x)][66:1-(x)])],401.111816,7
[67:1-(x)],403.421600,7
[67:2-([67:0-(x)][67:1-(x)])],407.128662,7
[73:0-(x)],408.184204,7
[68:1-(x)],408.902985,7
[68:2-([68:0-(x)][68:1-(x)])],411.874603,7

[74:0-(x)],412.390594,7
[69:1-(x)],414.924500,7
[69:2-([69:0-(x)][69:1-(x)])],418.575653,7
[75:0-(x)],419.567322,7
[70:1-(x)],421.118256,7
[70:2-([70:0-(x)][70:1-(x)])],425.066071,7
[76:0-(x)],425.196350,7
[71:1-(x)],427.101166,7
[71:2-([71:0-(x)][71:1-(x)])],431.357483,7
[77:0-(x)],431.453247,7
[72:1-(x)],433.107513,7
[72:2-([72:0-(x)][72:1-(x)])],437.399689,7
[78:0-(x)],437.587982,7
[73:1-(x)],439.568542,7
[73:2-([73:0-(x)][73:1-(x)])],444.121826,7
[79:0-(x)],444.387909,7
[74:1-(x)],445.540558,7
[80:0-(x)],449.006714,9
[74:2-([74:0-(x)][74:1-(x)])],449.241150,7
[75:1-(x)],451.721344,7
[81:0-(x)],454.820557,9
[75:2-([75:0-(x)][75:1-(x)])],455.534943,7
[76:1-(x)],457.781708,7
[82:0-(x)],461.547455,9
[76:2-([76:0-(x)][76:1-(x)])],462.005737,7
[77:1-(x)],464.545929,7
[77:2-([77:0-(x)][77:1-(x)])],466.984894,7
[83:0-(x)],468.234283,7
[78:1-(x)],470.863220,7
[84:0-(x)],474.641541,9
[78:2-([78:0-(x)][78:1-(x)])],475.209290,7
[79:1-(x)],476.427917,7
[85:0-(x)],479.938507,9
[79:2-([79:0-(x)][79:1-(x)])],480.164734,7
[80:1-(x)],482.226379,7
[80:2-([80:0-(x)][80:1-(x)])],484.530731,7
[86:0-(x)],485.507904,7
[81:1-(x)],488.489410,7
[81:2-([81:0-(x)][81:1-(x)])],492.441864,7
[87:0-(x)],492.632263,7
[82:1-(x)],494.277527,7
[82:2-([82:0-(x)][82:1-(x)])],497.976715,7
[88:0-(x)],498.427673,7

[83:1-(x)],499.980133,7
[89:0-(x)],503.897491,9
[83:2-([83:0-(x)][83:1-(x)])],504.272644,7
[84:1-(x)],506.960724,7
[90:0-(x)],510.719849,9
[84:2-([84:0-(x)][84:1-(x)])],511.481750,7
[85:1-(x)],512.848206,7
[85:2-([85:0-(x)][85:1-(x)])],515.157654,7
[91:0-(x)],517.081665,7
[86:1-(x)],519.240662,7
[86:2-([86:0-(x)][86:1-(x)])],522.915222,7
[92:0-(x)],523.553040,7
[87:1-(x)],524.380554,7
[93:0-(x)],528.267578,9
[87:2-([87:0-(x)][87:1-(x)])],528.558899,7
[88:1-(x)],529.832153,7
[88:2-([88:0-(x)][88:1-(x)])],533.554199,7
[94:0-(x)],534.481995,7
[89:1-(x)],535.463013,7
[95:0-(x)],539.300293,9
[89:2-([89:0-(x)][89:1-(x)])],539.576599,7
[90:1-(x)],541.695801,7
[90:2-([90:0-(x)][90:1-(x)])],545.478760,7
[96:0-(x)],545.718140,7
[91:1-(x)],547.317627,7
[91:2-([91:0-(x)][91:1-(x)])],549.613159,7
[97:0-(x)],552.171875,7
[92:1-(x)],553.848022,7
[92:2-([92:0-(x)][92:1-(x)])],557.945618,7
[98:0-(x)],558.210327,7
[93:1-(x)],560.126892,7
[93:2-([93:0-(x)][93:1-(x)])],563.038269,7
[99:0-(x)],564.085938,7
[94:1-(x)],566.660889,7
[94:2-([94:0-(x)][94:1-(x)])],569.230652,7
[100:0-(x)],570.150024,7
[95:1-(x)],573.005127,7
[95:2-([95:0-(x)][95:1-(x)])],576.729675,7
[101:0-(x)],577.069397,7
[96:1-(x)],579.006714,7
[96:2-([96:0-(x)][96:1-(x)])],581.586853,7
[102:0-(x)],583.060059,7
[97:1-(x)],585.816223,7

[97:2-([97:0-(x)][97:1-(x)])],589.605957,7
[103:0-(x)],590.060669,7
[98:1-(x)],591.849487,7
[98:2-([98:0-(x)][98:1-(x)])],595.687683,7
[104:0-(x)],596.068298,7
[99:1-(x)],597.328430,7
[99:2-([99:0-(x)][99:1-(x)])],599.429688,7
[105:0-(x)],601.298279,7
[100:1-(x)],603.438416,7
[106:0-(x)],607.638855,9
[100:2-([100:0-(x)][100:1-(x)])],608.241516,7

# APPENDIX L

**Table 7-0-1. Validation Testing Simulation RunTimes.**

| MAS I | | MAS II | | MAS III | |
|---|---|---|---|---|---|
| ASAE | ARENA | ASAE | ARENA | ASAE | ARENA |
| 312.579 | 315.804 | 304.952 | 304.18243 | 76.063 | 78.0595308 |
| 310.808 | 307.1639 | 301.26 | 307.245933 | 72.3679 | 76.7772477 |
| 312.874 | 315.8822 | 302.122 | 303.972975 | 73.7778 | 73.0953167 |
| 316.767 | 308.4117 | 305.925 | 304.596521 | 74.7071 | 78.8517371 |
| 314.227 | 310.7508 | 302.115 | 301.101406 | 74.403 | 73.2975925 |
| 313.344 | 307.0328 | 307.119 | 296.932721 | 73.1467 | 74.2853861 |
| 310.486 | 321.3813 | 303.732 | 306.84121 | 71.5561 | 76.2168015 |
| 312.369 | 310.8278 | 303.525 | 305.216856 | 74.629 | 75.6955142 |
| 311.05 | 314.4221 | 306.252 | 306.858818 | 75.3185 | 74.6217654 |
| 319.363 | 311.401 | 303.604 | 299.771587 | 73.0715 | 73.6028666 |
| 314.144 | 309.0963 | 308.538 | 304.151481 | 73.2496 | 75.69827 |
| 311.063 | 306.5036 | 305.989 | 304.815043 | 78.2903 | 76.4046019 |
| 314.023 | 311.8146 | 305.373 | 304.302262 | 75.7487 | 72.5881573 |
| 309.219 | 309.4241 | 306.408 | 307.59949 | 76.185 | 72.9803179 |
| 318.114 | 315.2398 | 304.979 | 306.098181 | 77.4499 | 75.0780898 |
| 318.914 | 314.2706 | 305.662 | 309.167806 | 72.3065 | 74.7978683 |
| 310.62 | 313.1828 | 306.448 | 303.126401 | 75.284 | 74.4119546 |
| 311.491 | 311.0907 | 302.632 | 302.003291 | 71.5092 | 79.9296554 |
| 313.593 | 313.1226 | 308.946 | 301.778218 | 73.7644 | 79.5608387 |
| 310.835 | 317.1965 | 303.405 | 297.740678 | 77.0385 | 74.2090786 |
| 317.563 | 313.2943 | 303.155 | 300.283716 | 74.5895 | 74.6427786 |
| 312.352 | 314.1194 | 304.024 | 305.162731 | 75.1935 | 79.5495275 |
| 310.438 | 310.1546 | 303.274 | 304.907468 | 75.1812 | 75.7994471 |
| 313.808 | 320.6581 | 305.738 | 305.304864 | 73.3728 | 75.9552784 |
| 313.324 | 305.3604 | 299.142 | 304.509008 | 76.2416 | 74.5615554 |
| 310.424 | 316.1447 | 306.066 | 303.207317 | 75.5897 | 73.2451216 |
| 311.968 | 309.3242 | 305.919 | 304.35925 | 76.7161 | 74.0182644 |
| 310.918 | 311.1759 | 308.819 | 298.529315 | 73.0823 | 73.0233466 |
| 310.413 | 317.6459 | 303.729 | 305.321305 | 72.3587 | 73.6094198 |
| 314.299 | 312.8097 | 304.152 | 297.291621 | 74.2565 | 76.9014888 |

| | | | | | |
|---|---|---|---|---|---|
| 314.015 | 311.4339 | 303.786 | 302.642353 | 73.8125 | 73.3847426 |
| 312.801 | 313.03 | 301.056 | 303.711248 | 76.2733 | 73.9238099 |
| 316.522 | 314.1213 | 303.324 | 307.558042 | 75.6447 | 73.9910993 |
| 310.575 | 309.9571 | 299.027 | 305.028064 | 71.9926 | 74.372536 |
| 311.26 | 315.1636 | 303.547 | 307.148748 | 71.3991 | 74.7667985 |
| 318.83 | 311.5781 | 306.001 | 301.397654 | 71.0814 | 74.5656937 |
| 314.248 | 316.1225 | 299.689 | 305.53517 | 75.9572 | 75.5603867 |
| 311.261 | 314.3065 | 302.614 | 304.367817 | 77.0012 | 70.0326021 |
| 312.969 | 307.281 | 305.481 | 301.505511 | 80.8566 | 72.7960568 |
| 314.729 | 313.8586 | 305.291 | 304.974861 | 76.1632 | 74.5544515 |
| 311.355 | 316.59 | 305.822 | 303.837815 | 74.0256 | 73.6804608 |
| 314.526 | 313.0955 | 307.653 | 300.342575 | 74.8286 | 75.285487 |
| 314.628 | 316.0004 | 300.875 | 301.868721 | 76.3056 | 73.2853839 |
| 309.97 | 311.0332 | 304.275 | 296.946182 | 75.3302 | 72.7326829 |
| 307.757 | 317.8728 | 303.921 | 301.616675 | 74.4542 | 72.2811364 |
| 313.351 | 313.2557 | 307.076 | 304.670164 | 74.4847 | 75.0609331 |
| 309.339 | 306.6591 | 302.151 | 301.415686 | 74.7667 | 76.1179902 |
| 311.878 | 312.9729 | 302.543 | 306.06585 | 76.609 | 76.0400748 |
| 314.531 | 315.6137 | 306.186 | 300.424427 | 72.2462 | 75.8714128 |
| 317.09 | 315.3124 | 301.454 | 302.486566 | 74.0043 | 74.3782542 |
| 313.629 | 309.9932 | 304.256 | 302.46205 | 74.0571 | 80.8909164 |
| 313.469 | 309.6521 | 308.173 | 304.124358 | 75.2402 | 73.8715772 |
| 316.368 | 314.3356 | 301.676 | 307.280533 | 75.9193 | 75.600761 |
| 311.777 | 311.4126 | 301.046 | 302.969125 | 75.1546 | 74.233101 |
| 309.777 | 308.7243 | 305.281 | 305.814367 | 74.9848 | 74.6070971 |
| 307.537 | 318.6329 | 304.757 | 309.83924 | 82.7475 | 72.0685527 |
| 317.374 | 315.4139 | 304.522 | 297.375985 | 74.1325 | 74.1779895 |
| 315.044 | 311.9877 | 303.655 | 298.183859 | 73.7326 | 72.9616713 |
| 315.442 | 305.5867 | 307.483 | 305.229738 | 72.6471 | 74.909775 |
| 317.576 | 320.8456 | 301.061 | 304.456291 | 76.4039 | 78.7662794 |
| 310.815 | 322.4258 | 301.393 | 299.94325 | 75.5435 | 78.4951485 |
| 314.112 | 315.934 | 301.572 | 300.150087 | 74.7371 | 73.843173 |
| 311.054 | 317.6597 | 308.49 | 306.028454 | 75.2945 | 75.3697694 |
| 310.899 | 314.0575 | 302.488 | 309.922713 | 73.2074 | 76.0873127 |
| 312.114 | 311.6158 | 299.261 | 304.530572 | 79.5659 | 74.2195447 |
| 313.71 | 315.3321 | 300.743 | 313.579933 | 73.9036 | 77.643758 |

| | | | | | |
|---|---|---|---|---|---|
| 312.026 | 315.6062 | 301.391 | 306.066501 | 72.5137 | 74.2117307 |
| 317.382 | 316.2448 | 296.692 | 305.84127 | 73.9926 | 77.1841787 |
| 305.782 | 311.2028 | 303.478 | 304.215314 | 76.5866 | 72.8546418 |
| 309.001 | 308.4612 | 300.382 | 300.614999 | 75.3679 | 70.7815889 |
| 309.136 | 313.9571 | 308.99 | 302.841878 | 74.8051 | 72.5641186 |
| 316.945 | 310.5779 | 298.721 | 302.074243 | 75.9102 | 74.5599559 |
| 309.454 | 310.9707 | 300.687 | 301.694671 | 74.0643 | 73.62881 |
| 312.325 | 312.7774 | 303.196 | 303.325393 | 74.091 | 73.5230259 |
| 321.192 | 317.6355 | 301.699 | 304.355137 | 74.4714 | 72.1835384 |
| 313.449 | 316.297 | 307.124 | 300.481443 | 73.9272 | 72.1610573 |
| 312.891 | 311.3797 | 298.857 | 304.957861 | 76.4382 | 73.1360622 |
| 314.709 | 308.9386 | 306.337 | 305.733653 | 74.0561 | 74.5320388 |
| 308.675 | 308.0665 | 301.557 | 306.175909 | 71.969 | 73.9044841 |
| 317.646 | 311.6184 | 303.294 | 300.006966 | 72.9937 | 75.2911048 |
| 317.196 | 316.6289 | 304.357 | 310.408723 | 75.3557 | 72.2430799 |
| 312.998 | 313.6369 | 305.665 | 300.24802 | 72.047 | 72.782903 |
| 310.341 | 312.2826 | 299.76 | 305.830734 | 73.864 | 72.9007847 |
| 308.754 | 307.523 | 302.068 | 305.274157 | 74.8001 | 75.0063092 |
| 314.731 | 315.4822 | 303.733 | 307.770898 | 77.4965 | 77.0415212 |
| 315.374 | 318.1452 | 301.051 | 304.810165 | 73.5124 | 72.7606034 |
| 321.327 | 307.9578 | 299.055 | 299.007618 | 75.3078 | 75.6303939 |
| 309.394 | 307.9228 | 306.935 | 308.226313 | 73.1263 | 78.072002 |
| 314.446 | 312.897 | 300.727 | 303.096354 | 75.3133 | 75.8148716 |
| 312.989 | 310.3005 | 299.449 | 306.631212 | 70.2361 | 75.6579871 |
| 310.49 | 316.2807 | 304.745 | 304.136981 | 72.9047 | 74.7593438 |
| 316.3 | 322.9567 | 307.322 | 307.601469 | 75.7594 | 75.1982056 |
| 311.709 | 317.2682 | 308.133 | 303.215084 | 74.4727 | 76.3075345 |
| 316.983 | 308.1194 | 303.56 | 309.973416 | 76.0954 | 76.9642466 |
| 315.881 | 312.0737 | 308.531 | 299.536904 | 76.3195 | 75.1078483 |
| 317.049 | 312.8969 | 305.453 | 301.923165 | 72.4522 | 72.5415884 |
| 310.095 | 317.9141 | 305.771 | 303.391619 | 71.803 | 76.0768853 |
| 314.567 | 313.5734 | 302.849 | 299.942124 | 74.5679 | 73.8619865 |
| 309.061 | 316.0722 | 300.139 | 300.635714 | 76.3387 | 74.5511649 |
| 307.094 | 307.4427 | 303.167 | 301.970345 | 72.3786 | 75.7956088 |

# APPENDIX M

## ASAE Source Code

```
//
//  main.cpp
//  ASAE (Automated Simulation Analysis Engine)
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: Simulation framework that creates a simulation from a text based definition.
//  the program parses the file and creates the model. The simulation is then run producing
//  an event log of the trial run.

#include <iostream>
#include "Simulation.hpp"
#include "DataCrawler.hpp"
#include <fstream>
#include <stdexcept>
#include <cctype>


enum log{
  NO_VERBOSE,
  VERBOSE
};
//************************************************************
//Function prototypes
int indexOfClosingBracket(std::string line);
std::vector<processInfo> getModelDefinition( int* numJobs,std::string);
void printModelDef(std::vector<processInfo> model);
//************************************************************

//Description: Main entry point in the program
```

```
int main(int argc, const char * argv[]) {
  std::string fileName;
  if(argc>1){
    std::cout<<"Provided Model Name: "<<argv[1]<<"\n";
    fileName = argv[1];
  }
  else{
    std::cout<<"Using default model name of model.txt\n";
    fileName = "./model.txt";
  }
  //try block to encapsulate the simulation logic
  try {
    int numJobs;
    std::vector<processInfo> modelDef =
getModelDefinition(&numJobs,fileName);
    printModelDef(modelDef);
    Simulation mySim;
    mySim.constructModel(modelDef);
    mySim.printModel();
    mySim.init();
    std::cout<<"Running Simulation with "<<numJobs<<" Jobs\n";
    mySim.run(numJobs,NO_VERBOSE);
  } catch (const std::runtime_error& e) {
    std::cout<<"\nERROR in Simulation\n";
    std::cerr << e.what() << std::endl;
    return EXIT_FAILURE;
  }

  //try block to encapsulate the dataCrawler logic
  try{
    //create a dataCrawler
    DataCrawler myCrawler("starts.txt","Finish.txt");
    myCrawler.run();
  }catch (const std::runtime_error& e) {
    std::cout<<"\nERROR in data Crawler!\n";
    std::cerr << e.what() << std::endl;
    return EXIT_FAILURE;
  }
```

```
 return EXIT_SUCCESS;
}

//Description:parse the text file containing the model definition
std::vector<processInfo> getModelDefinition( int* numJobs,std::string
fileName){
 std::vector<processInfo> model;
 std::fstream myFile;
 myFile.open(fileName.c_str());
 if(!myFile.is_open()){
  throw std::runtime_error("ERROR: failed to open file!");
 }
 std::cout<<"Model file found.\n";

 std::string line;
 std::getline(myFile,line);
 std::getline(myFile,line);
 int close = indexOfClosingBracket(line);
 std::string val = line.substr(1,close);
 *numJobs = atoi(val.c_str());
 std::getline(myFile,line);
 close = indexOfClosingBracket(line);
 val = line.substr(1,close);
 int numProcesses = atoi(val.c_str());
 std::cout<<"Number of processes is "<<numProcesses<<std::endl;
 for(int i = 0;i<numProcesses;++i){
  processInfo info;
  std::string pTime;
  std::getline(myFile, pTime);
  std::getline(myFile, pTime);
  int ending = indexOfClosingBracket(pTime);
  info.processTime = pTime.substr(1,ending);
  std::string posType;
  std::getline(myFile, posType);
  ending = indexOfClosingBracket(posType);
  info.processPos = atoi(posType.substr(1,ending).c_str());
  std::string downStream;
  std::getline(myFile, downStream);
```

```
    ending = indexOfClosingBracket(downStream);
    info.downStream = downStream.substr(1,ending);
    std::string upStream;
    std::getline(myFile, upStream);
    ending = indexOfClosingBracket(upStream);
    info.upStream = upStream.substr(1,ending);
    model.push_back(info);
    std::getline(myFile, pTime);
    std::getline(myFile, pTime);
  }
  myFile.close();
  return model;
}

//Description:utility function used when parsing the model file to find the end of
the line
int indexOfClosingBracket(std::string line){
  int ending = 1;
  int length = 0;
  while(ending < line.length()){
    if(line[ending] == '>'){
      break;
    }
    length++;
    ending++;
  }
  return length;
}

//Description:prints the model that is to be created from model file
void printModelDef(std::vector<processInfo> model){
  for (int i = 0; i<model.size(); ++i) {
    std::cout<<"Process: "<<i<<"\n";
    std::cout<<"\tProcessTime: "<<model[i].processTime<<"\n";
    std::cout<<"\tPosition: "<<model[i].processPos<<"\n";
    std::cout<<"\tDownstream connections: "<<model[i].downStream<<"\n";
    std::cout<<"\tUpstream connections: "<<model[i].upStream<<"\n";
  }
```

```
}



//
//  Buffer.hpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: defines the class structure for the buffer object

#ifndef Buffer_hpp
#define Buffer_hpp

#include <stdio.h>
#include <queue>
#include "Event.hpp"

enum BufferState{
  FULL,
  EMPTY,
  SPACE_LEFT,
  CAN_PULL
};

//Description: class that represents the process buffer within the system
class Buffer{
private:
  std::queue<Event> queue;
public:
  int capacity;
  Event GetNext();
  void placeInBuffer(Event E);
  int getState();
  int getNumInQueue();
};
```

```
#endif /* Buffer_hpp */
//
//  Buffer.cpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: defines the implementatin of the buffer object

#include "Buffer.hpp"

//Description:return the state of the buffer by saying if it is full,empty or still has
space
int Buffer::getState(){
  if (queue.size() == capacity) {
    return FULL;
  }
  else if(queue.size()== 0){
    return EMPTY;
  }
  else{
    return SPACE_LEFT;
  }
}

int Buffer::getNumInQueue(){
  return (int)queue.size();
}

//Description:return the next event in the queue
Event Buffer::GetNext(){
  Event E = queue.front();
  queue.pop();
  return E;
}
```

```
//Description: put the event in the buffer queue
void Buffer::placeInBuffer(Event E){
  queue.push(E);
}
//
//  Event.hpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: describes the structure of the event entity

#ifndef Event_hpp
#define Event_hpp

#include <stdio.h>
#include <iostream>


enum EventType{
  PULL_BUFFER,
  PUSH_BUFFER,
  START,
  FINISH
};

typedef struct{
  int triggerEventType;
  int NextProcess;
  int eventType;
  std::string jobID;
  float nextTime;
  int previousBuffer;
  int timeAtProcess;
}nextEventInfo;
```

```
//Description: class used to contain information important for processing an event
in the system
class Event{
private:
  int processID;
  int timesAtCurrentState;
  std::string jobID;
  int eventType;
  float processTime;
public:
  int previousBuffer;
  Event(int pID, std::string jID,int eventType,float PTime,int
numAtCurrentState,int previousBuffer = -1);
  float getProcessTime();
  int getProcessID();
  std::string getJobID();
  void printEvent();
  int getEventType();
  int getTimesAtCurrentState();
};

//Description: utility class used to compare the simtime for events when placing
into the priority queue
class Compare {
public:
  bool operator()(Event &a, Event &b)
  {
    if (a.getProcessTime() > b.getProcessTime())
    {
      return true;
    }
    else
    {
      return false;
    }
  }
};
```

```
#endif /* Event_hpp */
//
//  Event.cpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: defines how the create an event and get the needed information

#include "Event.hpp"

//Description:constructor for an event to be processed in simulation
Event::Event(int pID, std::string jID,int eventType,float pTime,int
numAtCurrentState,int pBuffer){
  this->processID = pID;
  this->jobID = jID;
  this->eventType = eventType;
  this->processTime = pTime;
  this->previousBuffer = pBuffer;
  this->timesAtCurrentState = numAtCurrentState;
}

//Description:get the type of event (PUSH,PULL,START, FINISH)
int Event::getEventType(){
  return eventType;
}

//Description: get the simtime that the event was scheduled with
float Event::getProcessTime(){
  return processTime;
}

//Description: get the job id assiciated with the current event
std::string Event::getJobID(){
  return jobID;
}
```

```
//Description:get the process id associated with the current event
int Event::getProcessID(){
  return processID;
}

//Description: Helper to get the times at the same state
int Event::getTimesAtCurrentState(){
  return timesAtCurrentState;
}

//Description:utility function to print the details of the event to the console for
debug
void Event::printEvent(){
  std::cout<<"********Printing Event*********\n";
  std::cout<<"PID: "<<processID<<"\n";
  std::cout<<"JOB_ID: "<<jobID<<"\n";
  std::cout<<"PREVIOUS_BUFFER: "<<previousBuffer<<"\n";
  if (eventType == PUSH_BUFFER) {
    std::cout<<"EventType: PUSH_BUFFER \n";
  }
  if (eventType == PULL_BUFFER) {
    std::cout<<"EventType: PULL_BUFFER \n";
  }
  if (eventType == START) {
    std::cout<<"EventType: START \n";
  }
  if (eventType == FINISH) {
    std::cout<<"EventType: FINISH \n";
  }
  std::cout<<"Process Time: "<<processTime<<"\n";
}

//
//  Process.hpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
```

```
//
//  Description: defines the structure of the process object

#ifndef Process_hpp
#define Process_hpp

#include <stdio.h>
#include <iostream>
#include "Buffer.hpp"
#include <string>
#include <cmath>
#include <random>
#include <algorithm>

typedef struct{
  std::string processTime;
  int processPos;
  std::string downStream;
  std::string upStream;
}processInfo;

typedef struct{
  int processID;
  float percentage;
  int capacity;
}downStreamConnection;

typedef struct{
  int processID;
  int bufferIndex;
}upStreamConnection;

typedef struct{
  float percent;
  int index;
}selectionChance;

enum Dist{
```

```
 TRIANGULAR,
 NORMAL,
 UNIFORM,
 CONSTANT
};

enum ProcessType{
 FRONT,
 MIDDLE,
 TERMINAL
};

//Description: class that stores info pertaining to a process in the system
class Process{
private:
 int jobNum;
 int processID;
 int distType;//defines how the times are generated
 int processType;//where the process is in the line
 float average;
 float minimum;
 float upper;
 float constant;
 std::default_random_engine generator;
 std::normal_distribution<float> distribution;
 std::uniform_real_distribution<float> U_distribution;
 float uniformLower;
 float uniformUpper;
 float normalAverage;
 float normalStdDev;
public:
 Process(){
  jobNum = 1;
 }
 std::vector<Buffer> process_Buffers;
 std::vector<downStreamConnection> downStreamDependencies;
 std::vector<upStreamConnection> upStreamDependencies;
 int getNumUpStreamDependencies();
```

```
  int getNumDownStreamDependencies();
  void setProcessID(int id);
  int getProcessType();
  int getBufferIndexToPush();
  void setProcessType(int type);
  void setDistType(int type);
  void setProcessParameters(std::string);
  float getProcessingTimeFromDist();
  void setBufferCapacity(int val,int ind);
  void setUpstreamDependencies(std::string);
  void setDownstreamDependencies(std::string);
  void printProcessInfo();
  void placeEventInBuffer(Event E,int ind);
  Event getEventFromBuffer(int ind);
  int BufferState(int i);
  void printNumInBuffers();
  std::string getJobNum();
  void AddOneJob();
};

#endif /* Process_hpp */
//
//  Process.cpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: Implementation of the process object

#include "Process.hpp"

//Description:defines what type of distribution a process adheres to
(triangular,normal, uniform)
void Process::setDistType(int type){
  this->distType = type;
}
```

```
//Description: standard function to set the process ID
void Process::setProcessID(int id){
  this->processID = id;
}

//Description: standard function to return the type of process indicating where in
the line it is
int Process::getProcessType(){
  return this->processType;
}

//Description:prints the parameters of the current process
void Process::printProcessInfo(){
  std::cout<<"\tProcess ID: "<<this->processID<<"\n";
  if(this->distType == TRIANGULAR)std::cout<<"\tDist Type:
TRIANGULAR\n";
  else if(this->distType == NORMAL)std::cout<<"\tDist Type: NORMAL\n";
  else if(this->distType == CONSTANT)std::cout<<"\tDist Type:
CONSTANT\n";

  if(this->processType == FRONT)std::cout<<"\tPos Type: FRONT\n";
  else if(this->processType == TERMINAL)std::cout<<"\tPos Type:
TERMINAL\n";
  else std::cout<<"\tPos Type: MIDDLE\n";

  std::cout<<"\tUpstream Dependencies: "<<this-
>upStreamDependencies.size()<<"\n";
  for(int i = 0;i<this->upStreamDependencies.size();++i){
    std::cout<<"\t\tProcessID: "<<upStreamDependencies[i].processID<<"\n";
    std::cout<<"\t\tBufferIndex: "<<upStreamDependencies[i].bufferIndex<<"\n";
  }
  std::cout<<"\tDownStream Dependencies: "<<this-
>downStreamDependencies.size()<<"\n";
  for(int i = 0;i<this->downStreamDependencies.size();++i){
    std::cout<<"\t\tProcessID: "<<downStreamDependencies[i].processID<<"\n";
    std::cout<<"\t\tPercentage: "<<downStreamDependencies[i].percentage<<"\n";
    std::cout<<"\t\tBuffer Capacity:
"<<downStreamDependencies[i].capacity<<"\n";
```

```
  }
}

//Description:create the dependencies that will be upstream from a process.
limited to 0-9
void Process::setUpstreamDependencies(std::string line){
  int num = std::atoi(line.substr(0,1).c_str());
  int start = 3;
  for(int i = 0;i<num;i++){
    //create each dependency
    upStreamConnection conn;
    conn.processID = std::atoi(line.substr(start,2).c_str());
    conn.bufferIndex = std::atoi(line.substr(start+3,1).c_str());
    upStreamDependencies.push_back(conn);
    start = start + 7;
  }
}

void Process::printNumInBuffers(){
  std::cout<<"Process ID "<<processID<<"\n";
  for(int i = 0;i<process_Buffers.size();++i){
    std::cout<<"Buffer Index "<<i<<" has
"<<process_Buffers[i].getNumInQueue()<<" in queue \n";
  }
}


//Description:create the downstream dependencies to control flow. limited to 0-9
void Process::setDownstreamDependencies(std::string line){
  int num = std::atoi(line.substr(0,1).c_str());
  int start = 2;
  float total = 0.0;
  for(int i = 0;i<num;i++){
    //create each dependency
    downStreamConnection conn;

    conn.processID = std::atoi(line.substr(start,2).c_str());
    conn.percentage = std::atof(line.substr(start+3,4).c_str());
```

```
    total = total + conn.percentage;
    conn.capacity = std::atoi(line.substr(start+8,2).c_str());
    if(conn.capacity<1||conn.capacity>99){
      throw std::runtime_error("\nBUFFER CAPACITY ERROR: Cannot have a
capacity less than 1 or greater than 99!\n");
    }
    downStreamDependencies.push_back(conn);
    Buffer buff;
    buff.capacity = conn.capacity;
    process_Buffers.push_back(buff);
    start = start +11;
  }

  if(abs(total-1.00) > 0.0001 && num != 0){
    throw std::runtime_error("\nDOWNSTREAM CONNECTION ERROR:
Downstream branching percentages must equal 1.00\n");
  }
}

//Description:return a random time for a triangular dist
float getTrianglarDistribution(float a, float b, float c)
{
  float randnum = (float)rand() / (float)RAND_MAX;
  float fc = (c - a) / (b - a);
  float val;
  if (randnum < fc && fc > 0.0)
  {
    val = a + sqrt(randnum*(b - a) / (c - a));
  }
  else
  {
    val = b - sqrt((1.0 - randnum)*(b - a) / (c - a));
  }
  return val;

}

//Description:bassed on the type of distribution get a random time
```

```
float Process::getProcessingTimeFromDist(){
 if (distType == TRIANGULAR) {
  return getTrianglarDistribution(minimum, upper, average);
 }
 else if(distType == NORMAL){
  return distribution(generator);
 }
 else if(distType == UNIFORM){
  return U_distribution(generator);
 }
 else if(distType == CONSTANT){
  return this->constant;
 }
 return 0.0;
}

//Description:set the type of process to indicate position in line
void Process::setProcessType(int type){
 this->processType = type;
}

//Description:standard setter for buffer capacity
void Process::setBufferCapacity(int val,int ind){
 this->process_Buffers[ind].capacity = val;
}


struct greater_than_key
{
 inline bool operator() (const selectionChance& struct1, const selectionChance&
struct2)
 {
  return (struct1.percent > struct2.percent);
 }
};

int Process::getBufferIndexToPush(){
 int numdep = (int)downStreamDependencies.size();
```

```
 std::vector<selectionChance> options;
 for(int i = 0;i<numdep;i++){
  selectionChance option;
  option.percent = downStreamDependencies[i].percentage;
  option.index = i;
  options.push_back(option);
 }
 std::sort(options.begin(), options.end(),greater_than_key());
 int ans = 0;
 int v2 = (rand() % 101);
 float percentR = (float)(v2)/100.0;
 float totalPercentage = 0.0;
 for(int i = 0;i<numdep;i++){
  totalPercentage = totalPercentage+options[i].percent;
  if(i == 0 && percentR <= totalPercentage){
   //select first
   ans = i;
   break;
  }
  else if(i == numdep-1){
   ans = i;
   break;
  }
  else{
   if(percentR<=totalPercentage){
    ans = i;
    break;
   }
  }
 }
 //std::cout<<"Selected index "<<ans<<" with probability of
"<<options[ans].percent<<"\n";
 return ans;
}

//Description: returns index of the process buffer downstream to place into
int Process::getNumDownStreamDependencies(){
 return (int)downStreamDependencies.size();
```

```
}

//Description: returns the amount of buffers feeding a process
int Process::getNumUpStreamDependencies(){
  return (int)upStreamDependencies.size();
}

//Description: takes an event and places in the process buffer
void Process::placeEventInBuffer(Event E,int ind){
  process_Buffers[ind].placeInBuffer(E);
}

//Description: takes an event fromt he process buffer
Event Process::getEventFromBuffer(int ind){
  return process_Buffers[ind].GetNext();
}

//Description: add one to the jobs complete parameter
void Process::AddOneJob(){
  jobNum++;
}

//Description: get the job number for the process
std::string Process::getJobNum(){
  return std::to_string(jobNum);
}

//Description: return the state of the process buffer (full,empty, space left)
int Process::BufferState(int i){
  return process_Buffers[i].getState();
}

//Description: set the timing parameters for distribution
void Process::setProcessParameters(std::string info){
  if (this->distType == TRIANGULAR) {
    //setParameters in process for triangular
    info.append(">");
    int done = 0;
```

```
  int index = 3;
  int front = 2;
  int num = 1;
  int length = 1;
  while(!done){
    if(info[index]=='>'){
      done =1;
      std::string max = info.substr(front,length);
      upper = atof(max.c_str());
      continue;
    }
    if(info[index]==':'){
      if(num ==1){
        std::string min = info.substr(front,length);
        minimum = atof(min.c_str());
        num++;
        length = 0;
        front = index+1;
        index++;
      }
      else if(num ==2){
        std::string avg = info.substr(front,length);
        average = atof(avg.c_str());
        num++;
        length = 0;
        front = index+1;
        index++;
      }
    }
    else{
      length++;
      index++;
    }
  }
}
else if(this->distType==NORMAL){
  //set for normal
  info.append(">");
```

```
    int done = 0;
    int index = 3;
    int front = 2;
    int num = 1;
    int length = 1;
    while(!done){
      if(info[index]=='>'){
        done =1;
        std::string std = info.substr(front,length);
        normalStdDev = atof(std.c_str());
        continue;
      }
      if(info[index]==':'){
        if(num == 1){
          std::string avg = info.substr(front,length);
          normalAverage = std::atof(avg.c_str());
          num++;
          length = 0;
          front = index+1;
          index++;
        }
      }
      else{
        length++;
        index++;
      }
    }
    std::normal_distribution<float>
mydistribution(normalAverage,normalStdDev);
    distribution = mydistribution;
  }
  else if(this->distType==CONSTANT){
    //set for constant
    //setParameters in process for constant
    info.append(">");
    this->constant = std::atof(info.substr(2,info.length()-2).c_str());
  }
  else if(this->distType==UNIFORM){
```

```
    //set for Uniform
    info.append(">");
    int done = 0;
    int index = 3;
    int front = 2;
    int num = 1;
    int length = 1;
    while(!done){
      if(info[index]=='>'){
        done =1;
        std::string std = info.substr(front,length);
        uniformUpper = atof(std.c_str());
        continue;
      }
      if(info[index]==':'){
        if(num == 1){
          std::string avg = info.substr(front,length);
          uniformLower = std::atof(avg.c_str());
          num++;
          length = 0;
          front = index+1;
          index++;
        }
      }
      else{
        length++;
        index++;
      }
    }
    std::uniform_real_distribution<float>
myUdistribution(uniformLower,uniformUpper);
    U_distribution = myUdistribution;
  }
}



//
```

```
//  Simulation.hpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: defines the use and structure of the simulation class

#ifndef Simulation_hpp
#define Simulation_hpp

#include <stdio.h>
#include <queue>
#include <vector>
#include <algorithm>
#include "Event.hpp"
#include "Process.hpp"
#include <iostream>
#include <fstream>
#include <iomanip>
#include <unordered_set>
#include <sstream>
#include <ctime>



//Description: main class that runs the simulation
class Simulation{
private:
  int debug;
  std::ofstream fileTimes;
  std::fstream startFile;
  std::fstream finishFile;
  std::fstream resultsFile;
  int finished;
  int jobsComplete;
  int jobsInSystem;
```

```
  int createJobID;
  float simTime;
  float timeStep;
  std::priority_queue <Event, std::vector<Event>, Compare> eventQueue;
  Process* simProcesses;
  int numProcesses;
public:
  int startRecordRow;
  int FinishRecordRow;
  Simulation();
  ~Simulation();
  int constructModel(std::vector<processInfo> &processes);
  int getFeedBufferState(Process P);
  void run(int numJobs, int verbose);
  void init();
  void printModel();
  void processNextEvent();
  nextEventInfo processCurrentEvent(Event current, int Process);
  void checkIfFinished(int);
  int getNumComponents(int);
  int getNumberOfEnterPoints(int processID);
};

#endif /* Simulation_hpp */

//
//  Simulation.cpp
//  ASAE
//
//  Created by Benjamin G Fields on 4/2/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//
//  Description: implementation of the simulation class

#include "Simulation.hpp"
```

```
//Description:constructor to create the simulation object
Simulation::Simulation(){
  simTime = 0.0;
  startFile.open("starts.txt", std::ios::out);
  finishFile.open("Finish.txt", std::ios::out);
  resultsFile.open("Results.txt",std::ios::out);
  if(!startFile.is_open()||!finishFile.is_open()){
    throw std::runtime_error("ERROR: Could not open Start and Finish log
files!");
  }
  jobsComplete = 0;
  jobsInSystem = 0;
  finished = 0;
  createJobID = 1;
  startRecordRow = 1;
  FinishRecordRow = 1;
  timeStep = 0.001;
  startFile<<"JobID Start,StartTime,Resource,JobsInSystem\n";
  finishFile<<"JobID END,ExitTime,JobsInSystem\n";
  srand((int)time(NULL));
}

//Description:destructor to save and close excel workbook
Simulation::~Simulation(){
  startFile.close();
  finishFile.close();
  std::cout<<"Terminating simulation\n";
}

//Description:takes the process info from the model file and creates the simulation
structure
int Simulation::constructModel(std::vector<processInfo> &processes){
  std::cout<<"Constructing the simulation model\n";
  numProcesses = (int)processes.size();
  simProcesses = new Process[numProcesses];
  for (int i = 0; i<numProcesses; ++i) {
    if(processes[i].processTime[0] == 'T'){
      if(this->debug)std::cout<<"Process "<<i<< " is Triangular \n";
```

```
      simProcesses[i].setDistType(TRIANGULAR);
    }
    else if(processes[i].processTime[0] == 'N'){
      if(this->debug)std::cout<<"Process "<<i<< " is Normal \n";
      simProcesses[i].setDistType(NORMAL);
    }
    else if(processes[i].processTime[0] == 'C'){
      if(this->debug)std::cout<<"Process "<<i<< " is Constant \n";
      simProcesses[i].setDistType(CONSTANT);
    }
    else if(processes[i].processTime[0] == 'U'){
      if(this->debug)std::cout<<"Process "<<i<< " is Uniform \n";
      simProcesses[i].setDistType(UNIFORM);
    }
    simProcesses[i].setProcessParameters(processes[i].processTime);
    simProcesses[i].setProcessType(processes[i].processPos);
    simProcesses[i].setUpstreamDependencies(processes[i].upStream);
    simProcesses[i].setDownstreamDependencies(processes[i].downStream);
    simProcesses[i].setProcessID(i);
  }
  return 0;
}

//Description:initializes the simulation with start jobs and begins each process to
wait for a job to arrive
void Simulation::init(){
  for (int i = 0; i< numProcesses; ++i) {
    if (simProcesses[i].getProcessType() == FRONT) {
      std::string id = "[" + simProcesses[i].getJobNum();
      id.append(":");
      id.append(std::to_string(i));
      id.append("-(x)]");
      Event E(i,id,START,simTime,0);
      eventQueue.push(E);
    }
    else{
      //schedule pull to start cycle
      Event E(i,"-1",PULL_BUFFER,simTime,0);
```

```
      eventQueue.push(E);
    }
  }
}

//Description:gets the state of the upstream buffers that feed a process
int Simulation::getFeedBufferState(Process proc){
  //check for pull
  if (proc.getNumUpStreamDependencies()==0) {
    return -1;
  }
  int state = EMPTY;
  for (int i = 0; i<proc.getNumUpStreamDependencies(); ++i) {
    int pos = proc.upStreamDependencies[i].processID;
    int buffIndex = proc.upStreamDependencies[i].bufferIndex;
    int buffState = simProcesses[pos].BufferState(buffIndex);
    if (buffState == EMPTY) {
      state = EMPTY;
      break;
    }
    else{
      state = CAN_PULL;
      continue;
    }
  }
  return state;
}

//Description: recursive helper function to determine how many components are
leaving system
int Simulation::getNumberOfEnterPoints(int processID){
  if(simProcesses[processID].getProcessType()==FRONT){
    return 1;
  }
  int num = 0;
  for(int i = 0;i<simProcesses[processID].getNumUpStreamDependencies();++i){
    int parent = simProcesses[processID].upStreamDependencies[i].processID;
    num = num + getNumberOfEnterPoints(parent);
```

```
  }
  return num;
}

//Description: return the number of components the part is composed of
int Simulation::getNumComponents(int current){
  //need to determine how many components are in the current job
  return getNumberOfEnterPoints(current);
}


//Description:process the event and then create the info for the next event to be
scheduled
nextEventInfo Simulation::processCurrentEvent(Event currentEvent, int
currentProcess){
  if(this->debug) std::cout<<"Process Type:
"<<simProcesses[currentProcess].getProcessType()<<std::endl;
  if(this->debug) currentEvent.printEvent();
  nextEventInfo info;
  std::string pid = std::to_string(currentProcess);
  std::string jid = currentEvent.getJobID();
  std::string resource = "worker "+pid;
  int FeedBufferState = getFeedBufferState(simProcesses[currentProcess]);
  if (this->debug) {
    if (FeedBufferState == EMPTY) {
      std::cout<<"FeedBufferState: EMPTY\n";
    }
    if (FeedBufferState == FULL) {
      std::cout<<"FeedBufferState: FULL\n";
    }
    if (FeedBufferState == CAN_PULL) {
      std::cout<<"FeedBufferState: CAN_PULL\n";
    }
    if (FeedBufferState == -1) {
      std::cout<<"FeedBufferState: NO BUFFER\n";
    }
  }
  //std::cout<<"\n\nBuffer check\n";
```

```
//simProcesses[currentProcess].printNumInBuffers();

//select the buffer to place into if need to push
int BuffertoPush;
if(currentEvent.previousBuffer != -1)
{
 BuffertoPush = currentEvent.previousBuffer;
}
else{
 if(simProcesses[currentProcess].getNumDownStreamDependencies()<2){
  BuffertoPush = 0;
  currentEvent.previousBuffer = 0;
 }
 else{
  BuffertoPush = simProcesses[currentProcess].getBufferIndexToPush();
  currentEvent.previousBuffer = BuffertoPush;
 }
}

int currentBufferState;
if(simProcesses[currentProcess].getNumDownStreamDependencies()==0){
 currentBufferState = -1;
}
else{
 //std::cout<<"trying to push into buffer "<<BuffertoPush<<"\n";
 currentBufferState = simProcesses[currentProcess].BufferState(BuffertoPush);
}

if(this->debug){
 if (currentBufferState == EMPTY) {
  std::cout<<"pushBufferState: EMPTY\n";
 }
 if (currentBufferState == FULL) {
  std::cout<<"pushBufferState: FULL\n";
 }
 if (currentBufferState == SPACE_LEFT) {
  std::cout<<"pushBufferState: SPACE_LEFT\n";
 }
```

```
   if (currentBufferState == -1) {
     std::cout<<"pushBufferState: NO BUFFER\n";
    }
  }

  if (currentEvent.getEventType() == PULL_BUFFER) {
    //try to pull

    //if there is a job then pull
    if (FeedBufferState == CAN_PULL) {
      //pull job from each buffer upstream and create new start with compound id
      std::string cid = "(";
      for (int i =0;
i<simProcesses[currentProcess].getNumUpStreamDependencies(); ++i) {
        int depend =
simProcesses[currentProcess].upStreamDependencies[i].processID;
        int buff =
simProcesses[currentProcess].upStreamDependencies[i].bufferIndex;
        Event E = simProcesses[depend].getEventFromBuffer(buff);
        cid.append(E.getJobID());
      }
      cid.append(")");
      std::string id = "[" + simProcesses[currentProcess].getJobNum();
      id.append(":");
      id.append(std::to_string(currentProcess));
      id.append("-");
      id.append(cid);
      id.append("]");

      info.eventType = START;
      info.jobID = id;
      info.NextProcess = currentProcess;
      info.nextTime = simTime;
      info.triggerEventType = PULL_BUFFER;
      info.previousBuffer = -1;
      info.timeAtProcess = 0;
    }
    else{
```

```
      //if no jobs then schedule another pull
      info.eventType = PULL_BUFFER;
      info.jobID = currentEvent.getJobID();
      info.NextProcess = currentProcess;
      info.triggerEventType = PULL_BUFFER;
      info.nextTime = simTime + timeStep;
      info.previousBuffer = -1;
      info.timeAtProcess = currentEvent.getTimesAtCurrentState()+1;
      if(info.timeAtProcess>1000000){
        std::string message = "DEADLOCK WARNING: Stuck at process
"+std::to_string(currentProcess)+" trying to pull!";
        throw std::runtime_error(message);
      }
    }
  }
  else if(currentEvent.getEventType() == PUSH_BUFFER){
    //try to push into buffer or wait if full

    //check if full
    if (currentBufferState != FULL) {
      //if not full then place into buffer and schedule pull unless at beginning you
schedule a start

      //push on buffer
      currentEvent.previousBuffer = -1;

simProcesses[currentProcess].placeEventInBuffer(currentEvent,BuffertoPush);
      if (simProcesses[currentProcess].getProcessType()==FRONT) {
        //schedule a start
        info.eventType = START;
        info.NextProcess = currentProcess;
        info.nextTime = simTime;
        std::string id = "[" + simProcesses[currentProcess].getJobNum();
        id.append(":");
        id.append(std::to_string(currentProcess));
        id.append("-(x)]");
        info.jobID = id;
        info.triggerEventType = PUSH_BUFFER;
```

```
        info.previousBuffer = -1;
        info.timeAtProcess = 0;
       }
       else{
        //schedule a pull
        info.eventType = PULL_BUFFER;
        info.NextProcess = currentProcess;
        info.nextTime = simTime;
        info.jobID = "-1";
        info.triggerEventType = PUSH_BUFFER;
        info.previousBuffer = -1;
        info.timeAtProcess = 0;
       }
      }
      else{
       //if full then schedule another push
       info.eventType = PUSH_BUFFER;
       info.jobID = jid;
       info.NextProcess = currentProcess;
       info.nextTime = simTime+timeStep;
       info.triggerEventType = PUSH_BUFFER;
       info.previousBuffer = BuffertoPush;
       info.timeAtProcess = currentEvent.getTimesAtCurrentState()+1;
       if(info.timeAtProcess>1000000){
        std::string message = "DEADLOCK WARNING: Stuck at process
"+std::to_string(currentProcess)+" trying to push!";
        throw std::runtime_error(message);
       }
      }
     }
     else if(currentEvent.getEventType() == START){
      if(simProcesses[currentProcess].getProcessType()==FRONT){
       jobsInSystem++;
      }
      //schedule finish event
      startFile<<jid.c_str()<<","
               <<std::to_string(currentEvent.getProcessTime()).c_str()<<","
               <<std::to_string(jobsInSystem).c_str()<<"\n";
```

```
    startRecordRow++;

    info.eventType = FINISH;
    info.jobID = currentEvent.getJobID();
    info.NextProcess = currentProcess;
    info.nextTime =
simTime+simProcesses[currentProcess].getProcessingTimeFromDist();
    info.triggerEventType = START;
    info.previousBuffer = -1;
    info.timeAtProcess = 0;

   }
   else if(currentEvent.getEventType() == FINISH){
    //try to push onto buffer or wait or nothing if terminal
    finishFile<<jid.c_str()<<","
    <<std::to_string(currentEvent.getProcessTime()).c_str()<<",";

    simProcesses[currentProcess].AddOneJob();
    if (simProcesses[currentProcess].getProcessType()==TERMINAL) {
     int numComponents = getNumComponents(currentProcess);
     //std::cout<<"Count of entrys is "<<numComponents<<"\n";
     jobsInSystem = jobsInSystem-numComponents;
     jobsComplete++;
     std::cout<<"\tJobs Complete: "<<jobsComplete<<"\n";
     std::cout<<"\tJobs in System: "<<jobsInSystem<<"\n";
     //check if at end of line if so then just assign -1 and return
     //schedule a pull
     info.eventType = PULL_BUFFER;
     info.jobID = "-1";
     info.NextProcess = currentProcess;
     info.nextTime = simTime;
     info.triggerEventType = FINISH;
     info.previousBuffer = -1;
     info.timeAtProcess = 0;
    }
    else{
     //if not terminal need to try to schedule push
     info.eventType = PUSH_BUFFER;
```

```
      info.jobID = jid;
      info.NextProcess = currentProcess;
      info.nextTime = simTime;
      info.triggerEventType = FINISH;
      info.previousBuffer = BuffertoPush;
      info.timeAtProcess = 0;
    }
    //write how many jobs are in the system
    finishFile<<std::to_string(jobsInSystem).c_str()<<"\n";
    FinishRecordRow++;
  }

  //return -1 -1 if no event is to be scheduled
  return info;
}

//Description:function that takes the next job in the queue and processes the event
void Simulation::processNextEvent(){
  if(this->debug) std::cout<<"\n**********PROCESSING
EVENT**********\n";
  int size = (int)eventQueue.size();
  if(this->debug) std::cout<<"Queue Size is "<<size<<"\n";
  if(this->debug) std::cout<<"Jobs in system: "<<jobsInSystem<<std::endl;
  if(this->debug) std::cout<<"Jobs complete: "<<jobsComplete<<std::endl;

  Event currentEvent = eventQueue.top();
  eventQueue.pop();
  int currentProcess = currentEvent.getProcessID();
  simTime = currentEvent.getProcessTime();
  nextEventInfo next = processCurrentEvent(currentEvent,currentProcess);

  if (next.NextProcess != -1) {
    Event
next_E(next.NextProcess,next.jobID,next.eventType,next.nextTime,next.timeAtP
rocess,next.previousBuffer);
    eventQueue.push(next_E);
  }
}
```

```
//Description:utility function to print out the simulation model as created in the
simulation
void Simulation::printModel(){
  std::cout<<"Printing Model \n";
  for (int i = 0; i<numProcesses; ++i) {
    std::cout<<"Process "<<i<<"\n";
    simProcesses[i].printProcessInfo();
  }
}

//Description:checks to see if the simulation has reached the terminated conditions
void Simulation::checkIfFinished(int num){
  if (jobsComplete == num) {
    //fileTimes.open ("test.csv", std::ofstream::out | std::ofstream::app);
    //fileTimes<<simTime<<"\n";
    finished = 1;
    //fileTimes.close();
    std::cout<<"\n**********DONE**********\n";
    std::cout<<"Simulation has reached finished state\n";
  }
  if (jobsComplete > num) {
    throw std::runtime_error("ERROR: Exceeded the number of jobs complete!");
  }
}

//Description:launch the simulaiton for a certain number of jobs
void Simulation::run(int numJobs, int verbose){
  this->debug = verbose;
  std::cout<<"\nBeginning Simulation with "<<numJobs<<" Jobs\n";
  while(!finished){
    processNextEvent();
    checkIfFinished(numJobs);
  }
  return;
}
//
//  DataCrawler.hpp
```

```
//  DataExtractor
//
//  Created by Benjamin G Fields on 5/15/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//

#ifndef DataCrawler_hpp
#define DataCrawler_hpp

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <stdexcept>
#include <unordered_set>
#include <sstream>
#include <chrono>
#include <iomanip>
#include <algorithm>

typedef struct{
  std::string jobID;
  float time;
  int jobsInSystem;
  std::vector<std::string> connections;
}Start;

typedef struct{
  std::string jobID;
  float time;
  int jobsInSystem;
}Finish;

typedef struct{
  float startTime;
  int Pid;
}startInfo;
```

```
typedef struct {
  int sum;
  float totaltime;
}processData;

class DataCrawler{
private:
  std::ifstream myStartFile;
  std::ifstream myFinishFile;
  std::ofstream resultsFile;
  std::vector<Start> myStarts;
  std::vector<Finish> myFinishs;
public:
  DataCrawler(std::string startFileName, std::string finishFileName);
  ~DataCrawler();
  std::vector<Start> readInStartsFile();
  std::vector<Finish> readInFinishFile();
  int getClosingBracket(int pos, std::string ID);
  int posOfNextBracket(std::string line, int start);
  int getPosOfColon(std::string line);
  int getPosOfDash(std::string line, int start);
  std::string getDependancy(std::string line);
  std::vector<std::string> getUpConnections(std::string line);
  int numJobsComplete(std::vector<Finish> &myFin,int pid);
  int getNumProcesses(std::vector<Finish> &myFin);
  void printTransitionStateMatrix(int** mat, int size);
  void printTransitionTimeMatrix(float** mat, int size);
  int getProcessID(std::string line);
  std::vector<int> getPIDS(std::string line);
  void countTransitions(int **mat, int size, std::vector<Finish> &myfin);
  startInfo getStart(std::string id,std::vector<Start> &myStart);
  void getTransitionTimes(float** mat, int size, std::vector<Finish> &myFin,
std::vector<Start> &myStart);
  void averageTransitionTimes(float** mat, int size,int** matFreq);
  int getMaxJobsInSystem(std::vector<Finish> &myFin);
  void getUtilizedBufferCapacity(int** bufMat,int** freqMat,int size);
  void getCapacityForPos(int** buffMat,int source, int destination);
  int getJobNum(std::string line);
```

```
 std::unordered_set<int> getNumberOfTerminalStates(int**freqMat,int size);
 std::vector<int> getProperStartConnections(int downstream,int upstream);
 void getAverageProcessTimes(float*mytimes, int size);
 float findStartTime(std::string id);
 processData countNumberOfTimesPIDFinishes(int id);
 void run();
};

#endif /* DataCrawler_hpp */
//
//  DataCrawler.cpp
//  DataExtractor
//
//  Created by Benjamin G Fields on 5/15/18.
//  Copyright © 2018 Benjamin G Fields. All rights reserved.
//

#include "DataCrawler.hpp"

//Description: Constructor to open and read files
DataCrawler::DataCrawler(std::string startFileName, std::string finishFileName){
 myStartFile.open(startFileName.c_str());
 if(!myStartFile.is_open()){
  std::cout<<"ERROR: Unable to open start event log\n";
  throw std::runtime_error("ERROR: failed to open starts.txt file!");
 }
 std::cout<<"Successfully opened start file\n";
 myFinishFile.open(finishFileName.c_str());
 if(!myFinishFile.is_open()){
  std::cout<<"ERROR: Unable to open finish event log\n";
  throw std::runtime_error("ERROR: failed to open finish.txt file!");
 }
 std::cout<<"Successfully opened finish file\n";
 resultsFile.open("Results.txt");
 if(!resultsFile.is_open()){
  std::cout<<"ERROR: Unable to open Results.txt\n";
  throw std::runtime_error("ERROR: failed to open Results.txt file!");
 }
```

```
  std::cout<<"Reading in Files\n";
  myStarts = readInStartsFile();
  myFinishs = readInFinishFile();
  if(myStarts.data() == NULL || myFinishs.data() == NULL){
    throw std::runtime_error("ERROR: Failed to create stored starts and
finishes!");
  }
  std::cout<<"Successfully read in start and finish files\n";
}

//Description:Destructor to close and present files
DataCrawler::~DataCrawler(){
  std::cout<<"Shutting down crawler\n";
  //clean up and present results
  myStartFile.close();
  myFinishFile.close();
  resultsFile.close();
  system("open -a TextEdit Results.txt");
}

//Description: Main runner to extract all process information from event data
void DataCrawler::run(){
  std::cout<<"\n\n*********Starting DataCrawler**********\n";
  std::cout<<"Number of Starts: "<<myStarts.size()<<"\n";
  std::cout<<"Number of Finishes: "<<myFinishs.size()<<"\n";

  auto now = std::chrono::system_clock::now();
  std::time_t time = std::chrono::system_clock::to_time_t(now);
  resultsFile<<"Results for "<<std::ctime(&time)<<"\n";
  std::cout<<"Results for "<<std::ctime(&time)<<"\n";

  resultsFile<<"Number of Start Records: "<<myStarts.size()<<"\n";
  resultsFile<<"Number of Finish Records: "<<myFinishs.size()<<"\n\n";

  float endTime = myFinishs.back().time;
  std::cout<<"Simulation runtime: "<<endTime<<"\n";
  resultsFile<<"Simulation runtime: "<<endTime<<"\n";
```

```
  int numProcesses = getNumProcesses(myFinishs);
  //create the transition state matrix
  int** transitionStateMatrix = new int*[numProcesses];
  for(int i =0;i<numProcesses;++i){
    transitionStateMatrix[i] = new int[numProcesses];
    for(int j = 0; j<numProcesses;++j){
      transitionStateMatrix[i][j] = 0;
    }
  }
  //find count of transitions
  countTransitions(transitionStateMatrix, numProcesses,myFinishs);
  std::unordered_set<int> numberOfTerminalStates =
getNumberOfTerminalStates(transitionStateMatrix,numProcesses);
  std::cout<<"\nThere are/is "<<numberOfTerminalStates.size()<<" Terminal
State(s)\n";
  resultsFile<<"\nThere are/is "<<numberOfTerminalStates.size()<<" Terminal
State(s)\n";
  int totalComplete = 0;
  for (auto itr = numberOfTerminalStates.begin(); itr !=
numberOfTerminalStates.end(); ++itr) {
    int numExit =numJobsComplete(myFinishs,*itr);
    totalComplete = totalComplete+numExit;
    std::cout<<"-Number of Jobs Exiting via PID "<<*itr<<": "<<numExit<<"\n";
    resultsFile<<"-Number of Jobs Exiting via PID "<<*itr<<":
"<<numExit<<"\n";
    std::cout<<"-Average time to produce one job:
"<<endTime/(float)numExit<<"\n\n";
    resultsFile<<"-Average time to produce one job:
"<<endTime/(float)numExit<<"\n\n";
  }
  std::cout<<"Total Complete Jobs: "<<totalComplete<<"\n";
  resultsFile<<"Total Complete Jobs: "<<totalComplete<<"\n";
  std::cout<<"Overall average time per job:
"<<endTime/(float)totalComplete<<"\n";
  resultsFile<<"Overall average time per job:
"<<endTime/(float)totalComplete<<"\n";

  int maxJIS = getMaxJobsInSystem(myFinishs);
```

```
  std::cout<<"Max Number of Components in System: "<<maxJIS<<"\n";
  resultsFile<<"\nMax Number of Components in System: "<<maxJIS<<"\n";


  std::cout<<"There are "<<numProcesses<<" processes\n";
  resultsFile<<"Number of Processes: "<<numProcesses<<"\n";

  //create the transition time matrix
  float** transitionTimeMatrix = new float*[numProcesses];
  for(int i =0;i<numProcesses;++i){
    transitionTimeMatrix[i] = new float[numProcesses];
    for(int j = 0; j<numProcesses;++j){
      transitionTimeMatrix[i][j] = 0.0;
    }
  }

  std::cout<<"Transition State Matrix\n";
  resultsFile<<"\nTransition State Matrix\n";
  //printTransitionStateMatrix(transitionStateMatrix, numProcesses);

  printTransitionStateMatrix(transitionStateMatrix, numProcesses);

  getTransitionTimes(transitionTimeMatrix, numProcesses, myFinishs, myStarts);
  std::cout<<"Transition Times summed\n";
  resultsFile<<"\nTotal Transition Times Matrix\n";
  printTransitionTimeMatrix(transitionTimeMatrix, numProcesses);
  averageTransitionTimes(transitionTimeMatrix, numProcesses,
transitionStateMatrix);
  std::cout<<"Average Transition Times averaged\n";
  resultsFile<<"\nAverage Transition Times Matrix\n";
  printTransitionTimeMatrix(transitionTimeMatrix, numProcesses);

  //create the buffer capacity matrix
  int** bufferCap = new int*[numProcesses];
  for(int i =0;i<numProcesses;++i){
    bufferCap[i] = new int[numProcesses];
    for(int j = 0; j<numProcesses;++j){
      bufferCap[i][j] = 0;
```

```
    }
  }

  getUtilizedBufferCapacity(bufferCap,transitionStateMatrix,numProcesses);
  std::cout<<"\nMax Utilized Buffer Capacity Matrix\n";
  resultsFile<<"\nMax Utilized Buffer Capacity Matrix\n";
  printTransitionStateMatrix(bufferCap, numProcesses);

  //create average process time array
  float* procTimes = new float[numProcesses];
  for(int i = 0;i<numProcesses;++i){
    procTimes[i]=0.0;
  }
  getAverageProcessTimes(procTimes, numProcesses);
  std::cout<<"\nAverage Process Times\n";
  resultsFile<<"\nAverage Process Times\n";
  for(int i = 0;i<numProcesses;++i){
    std::cout<<std::fixed<<std::setw(8)<<i<<" ";
    resultsFile<<std::fixed<<std::setw(8)<<i<<" ";
  }
  std::cout<<"\n";
  resultsFile<<"\n";
  for(int i = 0;i<numProcesses;++i){
    std::cout<<std::fixed<<std::setw(8)<<procTimes[i]<<"|";
    resultsFile<<std::fixed<<std::setw(8)<<procTimes[i]<<"|";
  }
  std::cout<<"\n";
  resultsFile<<"\n";
}

float DataCrawler::findStartTime(std::string id){
  for(int i=1;i<(int)myStarts.size();++i){
    if(myStarts[i].jobID.compare(id)==0){
      return myStarts[i].time;
    }
  }
  return 0.0;
}
```

```
processData DataCrawler::countNumberOfTimesPIDFinishes(int id){
  processData ans;
  ans.totaltime = 0.0;
  ans.sum = 0;
  for(int i = 1;i<(int)myFinishs.size();++i){
    std::string jid = myFinishs[i].jobID;
    if(getProcessID(jid)==id){
      float start = findStartTime(jid);
      //std::cout<<"JID "<<jid<<"starttime "<<start<<"\n";
      float duration = myFinishs[i].time - start;
      //std::cout<<"Duration of process "<<id<<" using jid "<<jid<<" is
"<<duration<<"\n";
      ans.totaltime+= duration;
      ans.sum++;
    }
  }
  return ans;
}

void DataCrawler::getAverageProcessTimes(float*mytimes, int size){
  for(int i = 0;i<size;++i){
    processData info = countNumberOfTimesPIDFinishes(i);
    mytimes[i] = info.totaltime/(float)info.sum;
  }
}

//Description: Helper function used to parse a jobID and get the job number
int DataCrawler::getJobNum(std::string line){
  int ans;
  int posOfColon = getPosOfColon(line);
  int length = posOfColon -1;
  ans = std::atoi(line.substr(1,length).c_str());
  return ans;
}

//Description: Helper function to retreive the list of proper jobNumbers related to
a downstream node connection
```

```
std::vector<int> DataCrawler::getProperStartConnections(int downstream,int
upstream){
  std::vector<int> ans;
  for(int i=1;i<(int)myFinishs.size();++i){
    if(getProcessID(myFinishs[i].jobID)==downstream){
      //std::cout<<"Found process id "<<downstream<<" in jobid
"<<myFinishs[i].jobID<<"\n";
      std::vector<std::string> depend = getUpConnections(myFinishs[i].jobID);
      for(int j = 0;j<depend.size();++j){
        //std::cout<<depend[j]<<"\n";
        if(depend[j]=="x"){
          continue;
        }
        int pid = getProcessID(depend[j]);
        if(pid == upstream){
          int jobNum = getJobNum(depend[j]);
          ans.push_back(jobNum);
        }
      }
    }
  }
  return ans;
}

//Description: Return how many exits to the system there are
std::unordered_set<int> DataCrawler::getNumberOfTerminalStates(int**
freqMat, int size){
  std::unordered_set<int> ans;
  for(int i = 0;i<size;++i){
    int sum = 0;
    for(int j = 0;j<size;++j){
      sum += freqMat[i][j];
    }
    if(sum==0){
      ans.insert(i);
    }
  }
  return ans;
```

```
}

//Description: For the position check what was the max capacity used for the
buffer
void DataCrawler::getCapacityForPos(int** buffMat,int upstream, int
downstream){
  // the source represents the finishs and the destination if the start
  //start with gap to see if there is a utilized buffer
  std::vector<int> properStartConnections =
getProperStartConnections(downstream, upstream);
  int startNum = 3;
  int bufferFound = 0;
  int finishNum = 1;
  int done = 0;
  int inBuffer = 0;
  int max = 0;
  int lastCheckedStartIndex = 0;
  int lastCheckedFinishIndex = 0;
  float startTime=0.0;
  float finishTime=0.0;
  while(!done){
    //start cheking for buffer case
    //find the relevent startjob with if of [startNum:destination-...
    int found1 = 1;
    for(int i = lastCheckedStartIndex;i<myStarts.size();++i){
      //if found then set found1 to 0 and break
      int jobNum = getJobNum(myStarts[i].jobID);
      int pid = getProcessID(myStarts[i].jobID);
      if(jobNum == properStartConnections[startNum-1] && pid == upstream){
        found1 = 0;
        lastCheckedStartIndex = i;
        startTime = myStarts[i].time;
      }
      if(found1==0)break;
    }
    int found2 = 1;
    for(int i = lastCheckedFinishIndex;i<myFinishs.size();++i){
      //if found then set found2 to 0 and break
```

```
    int jobNum = getJobNum(myFinishs[i].jobID);
    int pid = getProcessID(myFinishs[i].jobID);
    if(jobNum == finishNum && pid == downstream){
      found2 = 0;
      lastCheckedFinishIndex = i;
      finishTime = myFinishs[i].time;
    }
    if(found2==0)break;
  }
  if(found1==1||found2==1){
    done = 1;
    continue;
  }
  //use the times to check if there is a gap
  if(startTime < finishTime){
    //there is a buffer
    //need to add one
    inBuffer = inBuffer+1;
    if(inBuffer>max){
      max = inBuffer;
    }
    startNum++;
    bufferFound=1;
    continue;
  }
  else if(startTime >= finishTime && (startNum-finishNum)>2){
    inBuffer = inBuffer-1;
    finishNum++;
    continue;
  }
  else if(startTime >= finishTime && (startNum-finishNum)==2){
    startNum++;
    finishNum++;
    continue;
  }
 }
 //set the max seen
 buffMat[upstream][downstream] = max;
```

```
}

//Description: look through data to see what capacity of buffers was used
void DataCrawler::getUtilizedBufferCapacity(int** bufMat,int** freqMat,int
size){
  //for every position with a valid transition check if there is a utilized buffer
  for(int i = 0;i<size;++i){
    for(int j = 0;j<size;++j){
      //check everyposition valid transition
      if(freqMat[i][j]){
        //valid transition and need to check
        int upstream = i;
        int dowstream = j;
        getCapacityForPos(bufMat, upstream, dowstream);
      }
    }
  }
}

//Description: Helper function to get the closing bracket of an initial opening
bracket
int DataCrawler::getClosingBracket(int pos, std::string ID){
  int ans = -1;
  int ignore = 0;
  for(int i = pos;i<ID.length();++i){
    if(ID.substr(i,1)=="]" && ignore==0){
      ans = i;
      break;
    }
    else if(ID.substr(i,1)=="]" && ignore>0){
      ignore--;
    }
    else if(ID.substr(i,1)=="["){
      ignore++;
    }
  }
  return ans;
}
```

```
//Description: helper function used to find the next '[' character from a given
position
int DataCrawler::posOfNextBracket(std::string line, int start){
  int ans = -1;
  for(int i = start;i<(int)line.length();++i){
    if(line[i] == '['){
      ans = i;
      break;
    }
  }
  return ans;
}

//Description: find the position of the first ':' in the string
int DataCrawler::getPosOfColon(std::string line){
  int pos = -1;
  for(int i = 1;i<line.length();++i){
    if(line[i] == ':'){
      pos = i;
      break;
    }
  }
  return pos;
}

//Description: get the position of the first '-' in the string
int DataCrawler::getPosOfDash(std::string line, int start){
  int pos = -1;
  for(int i = start;i<line.length();++i){
    if(line[i] == '-'){
      pos = i;
      break;
    }
  }
  return pos;
}
```

```
//Description: extract the dependency string from the job id string
std::string DataCrawler::getDependancy(std::string line){
  std::string ans;
  int start = getPosOfColon(line);
  int end = getPosOfDash(line, start);
  int begin = end+2;
  int length = (int)line.length()-2-begin;
  ans = line.substr(begin,length);
  return ans;
}

//Description: get all dependencies within dependency string
std::vector<std::string> DataCrawler::getUpConnections(std::string line){
  std::vector<std::string> depend;
  std::string dependStr = getDependancy(line);
  if(dependStr=="x"){
    depend.push_back("x");
    return depend;
  }
  int index = 0;
  while(index != dependStr.length()-1){
    index = posOfNextBracket(dependStr, index);
    int closingPos = getClosingBracket(index+1, dependStr);
    std::string val = dependStr.substr(index,closingPos-index+1);
    depend.push_back(val);
    index = closingPos;
  }
  return depend;
}

//Description: read in the start events into memory for storage
std::vector<Start> DataCrawler::readInStartsFile(){
  std::vector<Start> readStarts;
  std::string line;
  int lineNum = 0;
  while(std::getline(myStartFile,line)){
    Start aStart;
    std::stringstream ss(line);
```

```
    std::string token;
    std::getline(ss,token, ',');
    aStart.jobID = token;
    if(lineNum != 0){
      aStart.connections = getUpConnections(token);
    }
    std::getline(ss,token, ',');
    aStart.time = std::atof(token.c_str());
    std::getline(ss,token, ',');
    aStart.jobsInSystem = std::atoi(token.c_str());
    readStarts.push_back(aStart);
    lineNum++;
  }
  return readStarts;
}

//Description: read in finish events into memory for storage and use
std::vector<Finish> DataCrawler::readInFinishFile(){
  std::vector<Finish> readFinish;
  std::string line;
  int lineNum = 0;
  while(std::getline(myFinishFile,line)){
    Finish aFinish;
    std::stringstream ss(line);
    std::string token;
    std::getline(ss,token, ',');
    aFinish.jobID = token;
    std::getline(ss,token, ',');
    aFinish.time = std::atof(token.c_str());
    std::getline(ss,token, ',');
    aFinish.jobsInSystem = std::atoi(token.c_str());
    readFinish.push_back(aFinish);
    lineNum++;
  }
  return readFinish;
}

//Description: return the number of jobs completed in simulation run from data
```

```
int DataCrawler::numJobsComplete(std::vector<Finish> &myFin,int pid){
 int max=0;
 for(int i=0;i<(int)myFin.size();++i){
  int Checkpid = getProcessID(myFin[i].jobID);
  if(Checkpid==pid){
   int num = getJobNum(myFin[i].jobID);
   if(num>max){
    max=num;
   }
  }
 }
 return max;
}

//Description: Return the number of unique processes in the system
int DataCrawler::getNumProcesses(std::vector<Finish> &myFin){
 std::unordered_set<int> myPIDs;
 for(int i = 1;i<(int)myFin.size();++i){
  std::string line = myFin[i].jobID;
  int start = getPosOfColon(line);
  int end = getPosOfDash(line, start);
  int length = end - start -1;
  int pid = std::atoi(line.substr(start+1,length).c_str());
  myPIDs.insert(pid);
 }
 return (int)myPIDs.size();
}

//Description:Print the transition frequency matrix to results and console
void DataCrawler::printTransitionStateMatrix(int** mat, int size){
 std::cout<<std::setw(3)<<"PID ";
 resultsFile<<std::setw(3)<<"PID ";
 for(int i = 0;i<size;++i){
  std::cout<<std::setw(3)<<i<<" ";
  resultsFile<<std::setw(3)<<i<<" ";
 }
 std::cout<<"\n";
 resultsFile<<"\n";
```

```
  for(int i = 0;i<size;++i){
    std::cout<<std::setw(3)<<i<<"|";
    resultsFile<<std::setw(3)<<i<<"|";
    for(int j = 0;j<size;++j){
      std::cout<<std::setw(3)<<mat[i][j]<<"|";
      resultsFile<<std::setw(3)<<mat[i][j]<<"|";
    }
    std::cout<<"\n";
    resultsFile<<"\n";
  }
}

//Description:Print the transition time matrix to results and console
void DataCrawler::printTransitionTimeMatrix(float** mat, int size){
  std::cout<<std::setw(9)<<std::right<<" PID";
  resultsFile<<std::setw(9)<<std::right<<" PID";
  for(int i = 0;i<size;++i){
    std::cout<<std::fixed<<std::setw(10)<<std::right<<i;
    resultsFile<<std::fixed<<std::setw(10)<<std::right<<i;
  }
  std::cout<<"\n";
  resultsFile<<"\n";
  for(int i = 0;i<size;++i){
    std::cout<<std::setw(9)<<std::right<<i<<"|";
    resultsFile<<std::setw(9)<<std::right<<i<<"|";
    for(int j = 0;j<size;++j){
      std::cout<<std::setfill('
')<<std::fixed<<std::right<<std::setw(9)<<std::setprecision(3)<<mat[i][j]<<"|";
      resultsFile<<std::setfill('
')<<std::fixed<<std::right<<std::setw(9)<<std::setprecision(3)<<mat[i][j]<<"|";
    }
    std::cout<<"\n";
    resultsFile<<"\n";
  }
}

//Description: Return the pid from a given jid string
int DataCrawler::getProcessID(std::string line){
```

```
  int start = getPosOfColon(line);
  int end = getPosOfDash(line, start);
  int length = end - start -1;
  int pid = std::atoi(line.substr(start+1,length).c_str());
  return pid;
}

//Description: get all the process IDs in a provided string
std::vector<int> DataCrawler::getPIDS(std::string line){
  std::vector<int> ans;
  int index = 0;
  while(index != line.length()-1){
    index = posOfNextBracket(line, index);
    int closingPos = getClosingBracket(index+1, line);
    int pid = getProcessID(line.substr(index+1,closingPos-index-1));
    ans.push_back(pid);
    index = closingPos;
  }
  return ans;
}

//Description: sum all transition counts in matrix
void DataCrawler::countTransitions(int **mat, int size, std::vector<Finish>
&myfin){
  for(int i = 1;i<myfin.size();++i){
    std::string jid = myfin[i].jobID;
    int pid = getProcessID(jid);
    //need to get upstream transitions
    std::string depend = getDependancy(jid);
    if(depend == "x")continue;
    //need to parse dependency to get the PID of each upstram task if not x
    std::vector<int> upstream = getPIDS(depend);
    for(int j = 0;j<upstream.size();++j){
      int process = upstream[j];
      mat[process][pid] = mat[process][pid]+1;
    }
  }
}
```

```
//Description: get the next start event info for a given finish job id
startInfo DataCrawler::getStart(std::string id,std::vector<Start> &myStart){
  startInfo info;
  for(int i=1;i<(int)myStart.size();++i){
    int depNum = (int)myStart[i].connections.size();
    for(int j = 0;j<depNum;++j){
      if(myStart[i].connections[j].compare(id)==0){
        //they are equal and return info
        info.startTime = myStart[i].time;
        info.Pid = getProcessID(myStart[i].jobID);
        return info;
      }
    }
  }
  info.startTime = -999.999;
  info.Pid = -99;
  return info;
}

//Description: Sum the transition times into the transition time matrix
void DataCrawler::getTransitionTimes(float** mat, int size, std::vector<Finish>
&myFin, std::vector<Start> &myStart){
  //for every finish except last position check when it begins next and is the major
dependency of the start
  for(int i = 1;i<(int)myFin.size()-1;++i){
    int pid = getProcessID(myFin[i].jobID);
    float finishTime = myFin[i].time;
    startInfo info = getStart(myFin[i].jobID,myStart);
    if(info.Pid <0)continue;
    float elapsedTime = info.startTime - finishTime;
    mat[pid][info.Pid] =  mat[pid][info.Pid] + elapsedTime;
  }
}

//Description: Use frequencies and summed amounts to provide an average
transition time matrix
void DataCrawler::averageTransitionTimes(float** mat, int size,int** matFreq){
```

```
  for(int i = 0;i<size;++i){
    for(int j = 0;j<size;++j){
      if(matFreq[i][j]==0)continue;
      mat[i][j] = mat[i][j]/(float)matFreq[i][j];
    }
  }
}

//Description: Return the max seen number of jobs in system from finish events
int DataCrawler::getMaxJobsInSystem(std::vector<Finish> &myFin){
  int max = myFin[0].jobsInSystem;
  for(int i = 1;i<(int)myFin.size();++i){
    if(myFin[i].jobsInSystem > max){
      max = myFin[i].jobsInSystem;
    }
  }
  return max;
}
```