12-2004

Web based evaluation of material handling alternatives for automated manufacturing : a parallel replications approach.

John B. Casebier 1973-
University of Louisville

Web Based Evaluation of Material Handling Alternatives for Automated Manufacturing: A Parallel Replications Approach

By

John B. Casebier
B.S.I.E. University of Louisville, 2003

A Thesis
Submitted to the faculty of the
Graduate School of the University of Louisville
In Partial Fulfillment of the Requirements
For the Degree of

Master of Engineering
In
Industrial Engineering

Department of Industrial Engineering
J.B. Speed School of Engineering
University of Louisville
Louisville, Kentucky

December 2004

Web Based Evaluation of Material Handling Alternatives for Automated Manufacturing: A Parallel Replications Approach

By

John B. Casebier
B.S.I.E., University of Louisville, 2003

A Thesis Approved on

December 10, 2004

By the Following Reading Committee

Dr. William E. Biles, Thesis Director

Dr. Gerald W. Evans

Dr. Rammohan K. Ragade

ABSTRACT

This research project describes the application of a master/slave configuration of processors to study a comparison of alternative material handling configurations for automated manufacturing. Such a study usually requires a large number of simulation replications, and carrying out those replications on a multi-processor platform can yield a significant savings in elapsed time. The objective of this study is to develop such a platform.

In the present application, a parallel replication simulation system is developed to foster simultaneous processing. This system utilizes two separate applications to facilitate communication between master and slave computers. Additionally, the "master" and "slave" or "client" applications work in conjunction with a specialized set of Arena™ simulation models.

The simulation models considered in this research effort represent two types of transport mechanisms in a closed cell work environment. Transport type, transport velocity, and an associated efficiency factor of a machine in cell 3. These models have been modified for use within the parallel replications environment.

This system uses $2^3 = 8$ design points (simulation models) for an experimental design. The models are transferred to remote PCUs via Transmission Control Protocol (TCP) file transfer. Following receipt, the models are executed, and results sent back to the "master" application where factor significance is determined. The resulting Metamodel (Kleijnen, 1979) describes a linear relationship between model variables, and system output.

ACKNOWLEDGEMENTS

This thesis represents a culmination of ideas and special interests; none of which could have been realized without the support and guidance provided by Dr. William E. Biles. For this reason, the author gratefully acknowledges Dr. Biles and thanks him for the education and knowledge gained through completing this thesis. The author also wishes to thank Dr. Gerald W. Evans and Dr. Rammohan K. Ragade for taking the time to serve on this reading committee. Additionally, the author wishes to extend special thanks towards his Aunt and Uncle, Jean and Bill Casebier. Their support and friendship significantly contributed to the author's success of this project and completion of his engineering degrees. Last but most importantly, the author wishes to thank his mother, Pat Snow, and wife Nazanin. For without their respective support and patience, this thesis may not have been realized.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

A. OVERVIEW

The need for fast and reliable simulation results is an ever-increasing topic within industry today. In the past, conducting simulation studies required vast resources and processing time to complete. Advancing technology is now helping us to do more in less time. Although we can complete simulation studies in a fraction of the time required in the past, complex simulations take greater amounts of time and processing power to complete. The goal of this study is to define a network based simulation engine, which will allow simulation replications to be completed in a fraction of the original simulation time requirements.

Large simulation models can sometimes take a week or more in order to run to completion (Anderson, 2004). Building, verifying, and validating a simulation model is sometimes a daunting task, however if we could run these models faster, the systems could be optimized faster as well. Although model optimization is currently possible with 'turn-key' commercial software, network and Metamodel (Kleijnen, 1979) optimization remains highly attractive.

Although similar research studies have been conducted using packages such as JavaSim (Luo, 2000), and other Java-based simulations (Marr, 2000), the objective of this study is to create and implement a variation of these attempts using Microsoft Visual Basic 6.0, and Rockwell Automation's Arena™.

B. FUNDAMENTALS OF SIMULATION

The fundamentals of simulation are based on basic probability and statistics. Although simulation does not seem very basic, the driving force behind all simulation is the "event", or the "chance" of something happening. Over time, if we measure or record events occurring around us a "pattern" of some form will emerge. A person may say that "there are no patterns in random events", and, they would be correct. However, over time a "trend" will start to emerge, and it is this trend that enables us to form relationships between systems.

Through the use of statistical tests, a person may hypothesize possible distributions of random occurrences. After this hypothesis has been tested and validated, the mathematical relationship can be formed. By using these mathematical expressions, we can create pseudo-random occurrences that follow our 'hypothesized' trend and mimic the actual scenario in which we wish to recreate. Although a true random generator has not yet been identified, it is possible to generate pseudo-random occurrences through formed mathematical relationships. The study of different random number generators merits research, but the validity or effectiveness of these generators is not the focus of this particular research effort.

Simulation is a very broad term that encompasses a set of problems or approaches to solving those problems. Simulations can be conducted on paper, computers, and in "real life", whether via spreadsheets or via simulation modeling languages. In short, a simulation represents a model of an actual process or situation. The general outcome desired from carrying out a simulation, is to better understand how a given system or group of systems interact with each other.

Simulations models are effective tools within many common industries such as manufacturing, distribution and logistics, medicine, the military, and public transportation systems. When you think about it, real-life experimentation with these types of systems can be quite expensive and very dangerous.

A large percentage of industries are turning to third-world service/product vendors for inexpensive products. At first, one would think that there are no problems in doing this, unless of course you consider the jobs that are lost from outsourcing in efforts to gain a less expensive product. If a company can reduce a product cost by half of its original value, then the attractiveness readily presents itself. However, some may not consider the amount of time required to transport the products from one side of the planet to the other. This transport time can be extremely influential on the amount of product that a manufacturer or distributor must carry in order to maintain high customer service levels.

Another area extremely sensitive to change is the medical industry. For example, what would happen if two doctors per shift were cut from an emergency room? Additionally, what would happen if the number of available ambulances was reduced by half? When put into perspective, it is easy to see the value of simulation.

C. PURPOSE

Again, the goal of this study is to construct and implement a simulation configuration that will support web-based or network based simulation systems. Although current technology allows us to simulate complex modeling scenarios, vast amounts of time can be required to complete adequate simulation trials in order to determine the steady-state operation of a proposed system. Through the use of parallel processors, multiple simulation replications can be executed in parallel at the same time. Although parallel and distributed simulation has been performed in the past, this study examines the effectiveness and applicability of utilizing readily available commercial software packages (Arena ™) to perform optimization and steady-state analysis.

CHAPTER II

PHYSICAL CONFIGURATION

A. SYSTEM ARCHITECTURE

Although there are several different methods in which to set up a network in order to facilitate network simulation, this system, the Parallel Replication Simulation Engine (PRSE), employs a "master-slave" relationship, (Biles, 1985). An array of "slave" computers is dedicated to execute parallel simulation replications, while the "master" is utilized as the "simulation engine". The "master" is configured for remote access in order to build (should the need arise) and execute simulation models in a network environment.

The PRSE system architecture is constructed in the following manner:

Hardware/Software:

Slave computers

- Dell Optiplex GX1
    - 128Mb RAM
    - 400MHz processor
- Network Interface Card
- Microsoft Windows XP Professional
- Arena 7.01 Student Edition
- Parallel Replication Client (PRC) Application

Master computer

- Dell Optiplex GX1
    1. 256Mb RAM
    2. 400Mhz processor.
- Network Interface Card
- Microsoft Windows XP Professional
- Microsoft Office XP Professional
- Arena 7.01 Student Edition
- Microsoft Visual Studio 6.0
- Parallel Replication Master (PRM) Application

Web Server

- Cobalt Qube 2
    - Serving as a Port Forwarder, and
    - Serving as a System Router

Switching Device

- Belkin OmniView 8-Port KVM Switch

Ethernet Hub

- 3Com OfficeConnect Dual Speed Switch 16 Plus

The physical simulation system configuration currently operates with up to eight "slaves", one "master", a web server, an 8-port switching device, and an Ethernet hub (Figure 2-1 demonstrates this relationship). The "master" computer communicates with the "slave array" by utilizing the TCP/IP (Transmission Control Protocol / Internet Protocol) construct. Connections are formed via TCP

interaction between computers and network interface cards (NICs), which are common to all functioning elements within the simulation array.

The Cobalt Qube™ (Sun Microsystems) functions with two NICs. These network cards work in conjunction with each other. Through this relationship, the Qube is able to serve as a port forwarder. By using Port Forwarding, a remote analyst may directly connect to the "master" via Remote Desktop Connection (Microsoft) from anywhere in the world (Figure 2-1). Because the Qube is a "turn-key" unit, its operating system supports a number of functions that would be found in a typical router; therefore it serves a dual purpose.

In order for a computer (whether array or web-based) to function as a 'slave', the Parallel Replication Client (PRC) application must be downloaded and installed on the local machine. The availability of this PRC application is afforded by the use of the Cobalt Qube™ and a web page devoted to hosting application files.

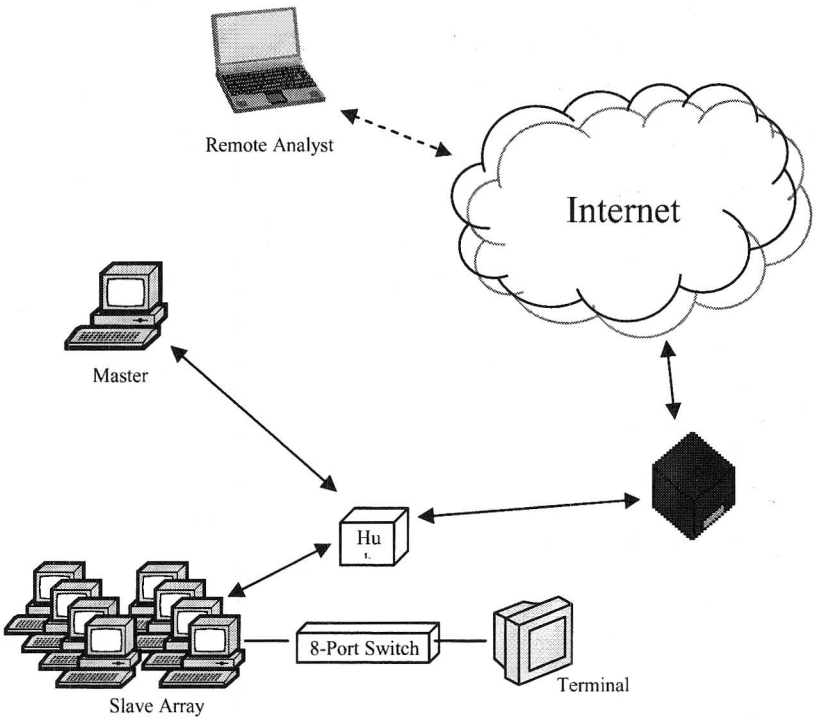


Figure 1-1: Physical representation of the Parallel Replication Simulation System (PRSS).

CHAPTER III

THE EXPERIMENTAL SIMULATION MODEL

A. MODEL SELECTION

As noted, the focus of this research is not a typical industrial simulation study, but rather an attempt to optimize the time to complete industrial simulation applications. For this reason, textbook examples of a multi-cellular manufacturing system were chosen from the third edition of Simulation with Arena (Kelton, Sadowski, and Sturrock, 2004).

The experimental simulation models consist of four manufacturing cells, each of which are preceded and followed by assignment modules (Figures 3-1, and 3-2). These assignment modules increment and decrement a variable tracking that cell's respective work-in-progress (WIP). Additionally, cells 1, 2, and 4 have only a single machine, while cell 3 has two machines (old and new respectively). Model 8-2 simulates free path transporters, while model 8-4 simulates a non-accumulating conveyor system (Kelton, Sadowski, and Sturrock, 2004). The difference in transport mechanisms influenced model selection for this study, as an evaluation of material handling alternatives is required.

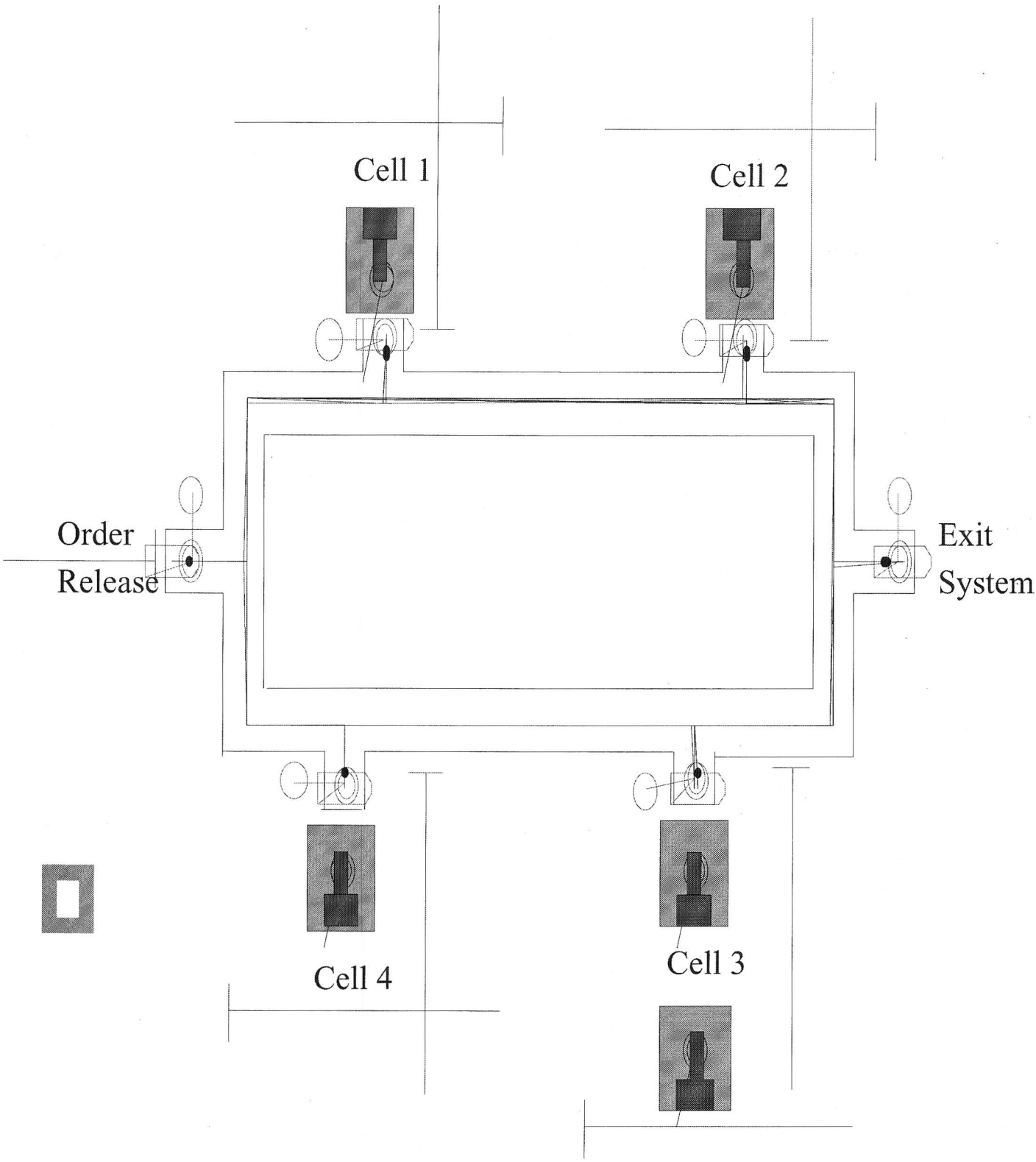


Figure 3-1: Physical layout of experimental simulation model (Kelton, Sadowski, and Sturrock, 2004).

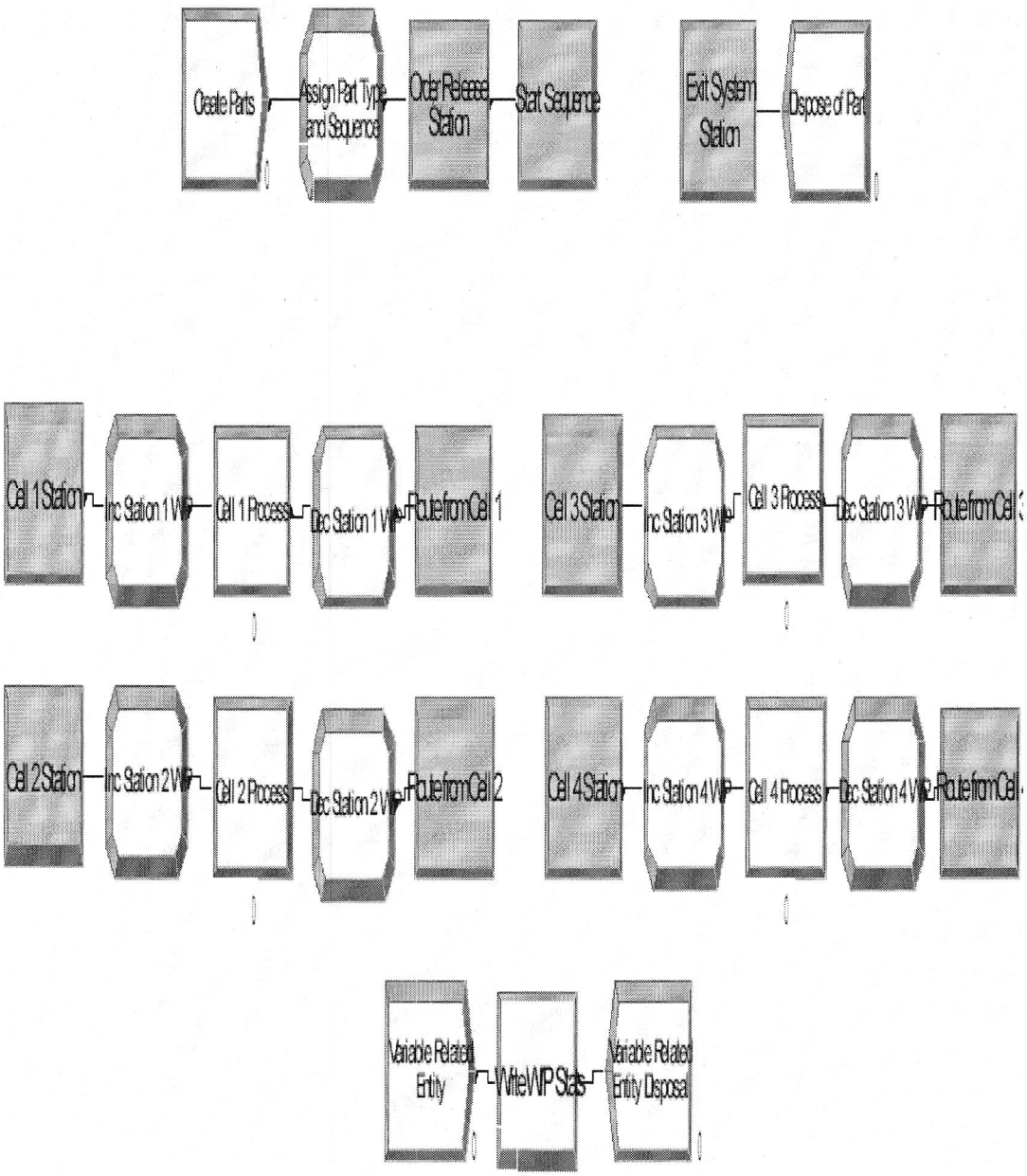


Figure 3-2: Arena model logic for experimental simulation model (Kelton, Sadowski, and Sturrock, 2004)

B. ASSIGNMENT MODULES

As highlighted, experimental factors used within this optimization model include transport type, transport velocity, and the efficiency of the old machine in work cell 3. As shown in Figure 3-2, Assignment Modules precede and follow processing stations. These assignment modules are used to define variables associated with WIP for each cell. The simulation variables, called "*Station(n)WIP*" (*n*, respective work cell), are incremented as shown in Figure 3-3, and decremented as shown in Figure 3-4. As entities pass through respective processing stations or work cells, the level of WIP is continuously monitored. At replication end, the maximum WIP (MaxWIP) out of all four work cells in considered and then written to text file for transmission to the PRM application. Equation 1 expresses this mathematical relationship.

Maximum WIP (MaxWIP) for cell(n):

$$Station(n)WIP = MAX(WIP(REPLICATION(n)) \quad (1)$$

Where, n = *Station 1, 2, 3, or 4.*

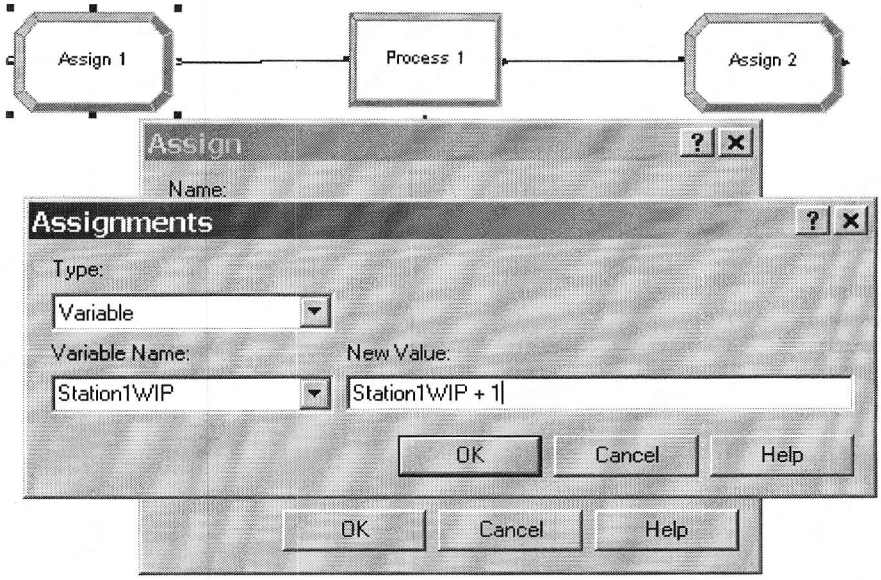


Figure 3-3: Assignment Module – Incremental Variable Definition

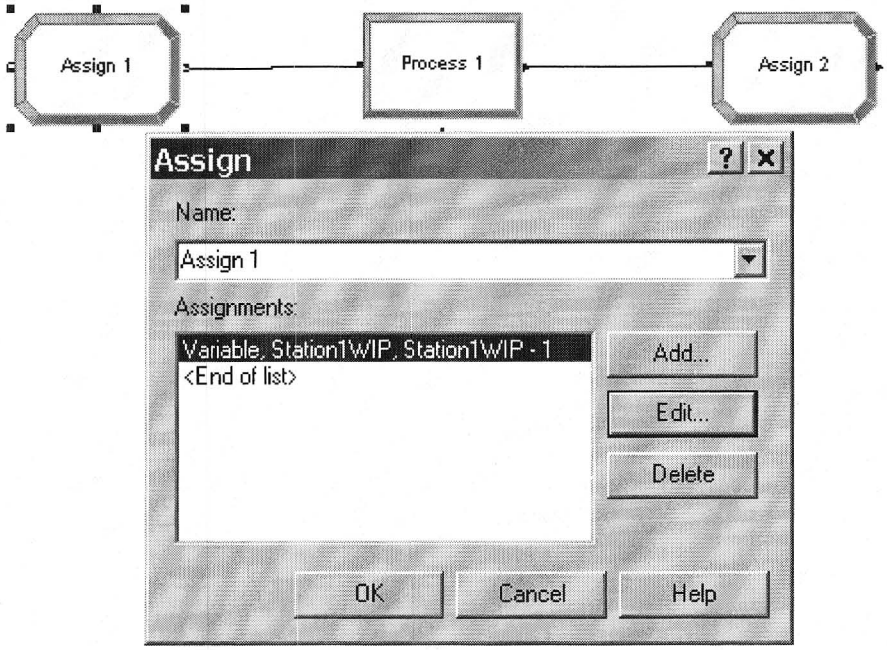


Figure 3-4: Assignment Module – Decremental Variable Definition.

C. READWRITE MODULE

Following the definition of the "*Station(n)WIP*" variable, a procedure to record the maximum WIP (MaxWIP) value must be defined. In order to record these values, an additional simulation group must be introduced to the experimental model(s) (Figure 3-5). To accomplish this task, three separate modules were introduced, Create, ReadWrite, and Dispose (respectively).

Figure 3-5: The Additional Simulation Group.

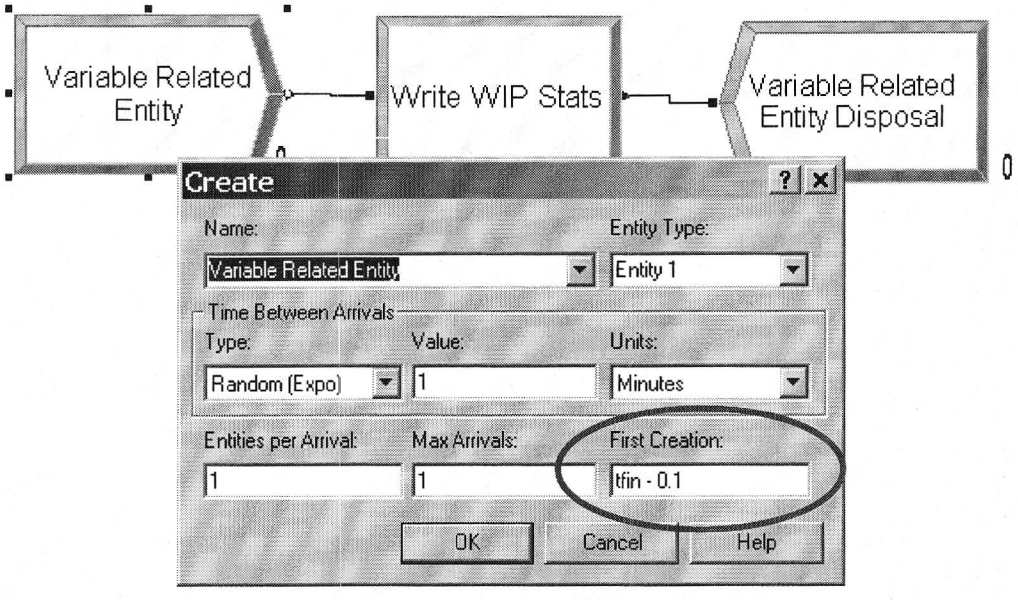


Figure 3-6: Special Entity Generation – The Final Entity.

The Create module introduces a new variable related entity, namely "The Final Entity", to the system slightly before the experimental module simulation ends. This entity creation is expressed in Equation 2, and also highlighted in Figure 3-6. Because this entity is virtually created at the instant that the experimental model ends, the inter-arrival distribution of this "final entity" is negligible.

The Final Entity:

$$Time = t - 0.1 \text{ m}inutes \tag{2}$$

Following the creation of the variable related entity, the maximum WIP value for any work cell in each replication must be written to a text file. This task is accomplished through the use of a ReadWrite module (Figure 3-7). The ReadWrite module, part of the advanced process template in arena, gives the simulation analyst the capability to specify what value will be recorded at time t-0.1 (time t represents the time at which the simulation is scheduled to end). Time t-0.1 is the minimum amount of time required in order for data to be recorded; time t-0.001 prevents data from being written. *As* previously stated, the ReadWrite module in the experimental model records the maximum WIP value for each of the four work cells (Kelton, Sadowski, and Sturrock, 2004) over each replication.

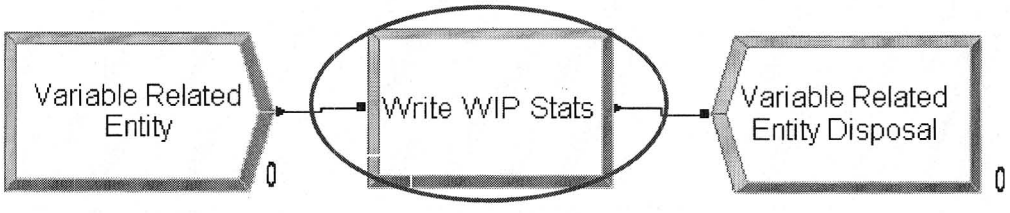


Figure 3-7: Special Entity Generation – Create, ReadWrite, and Dispose Modules.

1. EXPRESSION BUILDER

Through the use of an expression, the overall maximum WIP value is determined as shown in Figure 3-8, and defined by Equation 3. This expression must be built into the ReadWrite module, and is done so by utilizing built-in Math Functions in Arena ™. Along with defining the value to be recorded by the ReadWrite module, it is important to note that the assignment type is defined as "other", as shown in Figure 3-9.

$$MaxWIP = MX(Station1WIP, Station2WIP, Station3WIP, Station4WIP) \qquad (3)$$

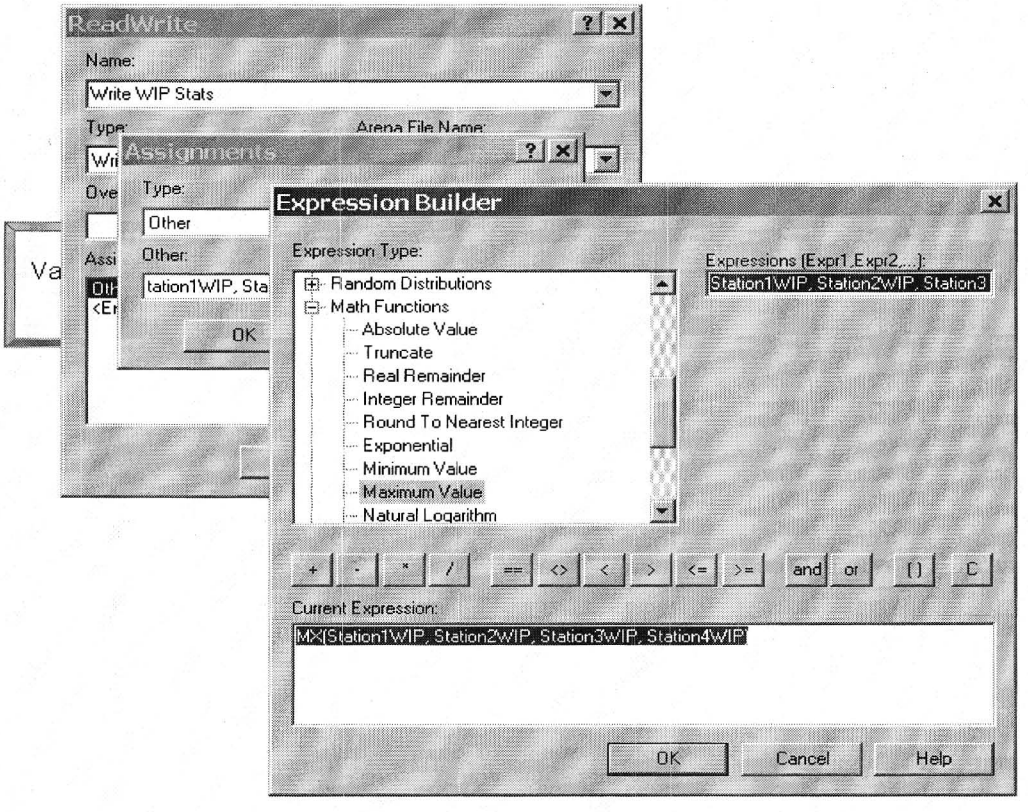


Figure 3-8: ReadWrite Expression Builder.

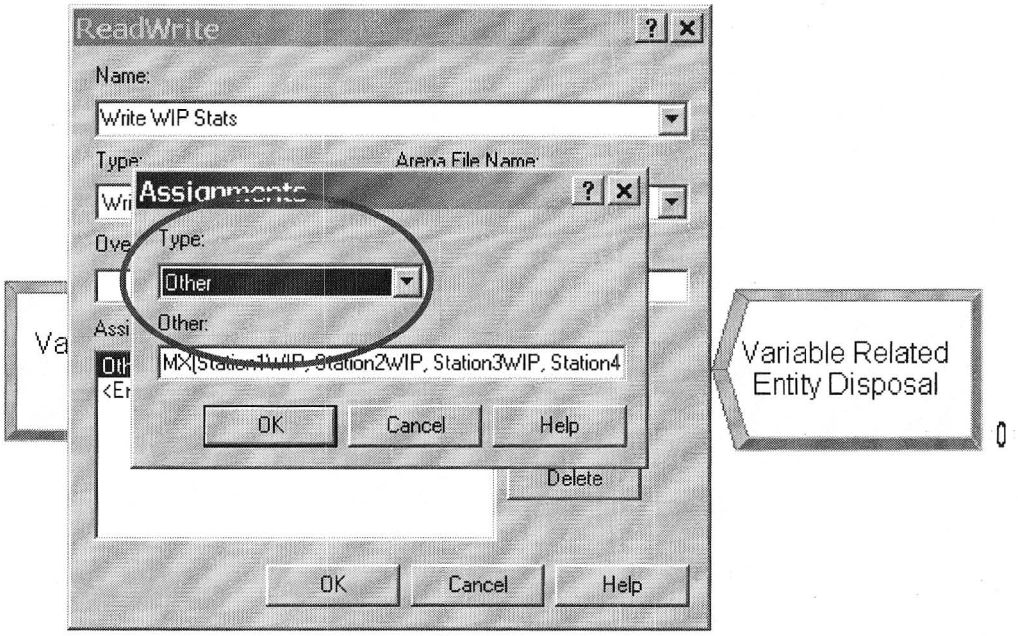


Figure 3-9: ReadWrite Assignment Type.

The next step that is required in recording variable data from Arena™ replications is to define the action of the ReadWrite module as well as to define the name of the file that is to be saved (Figure 3-10).

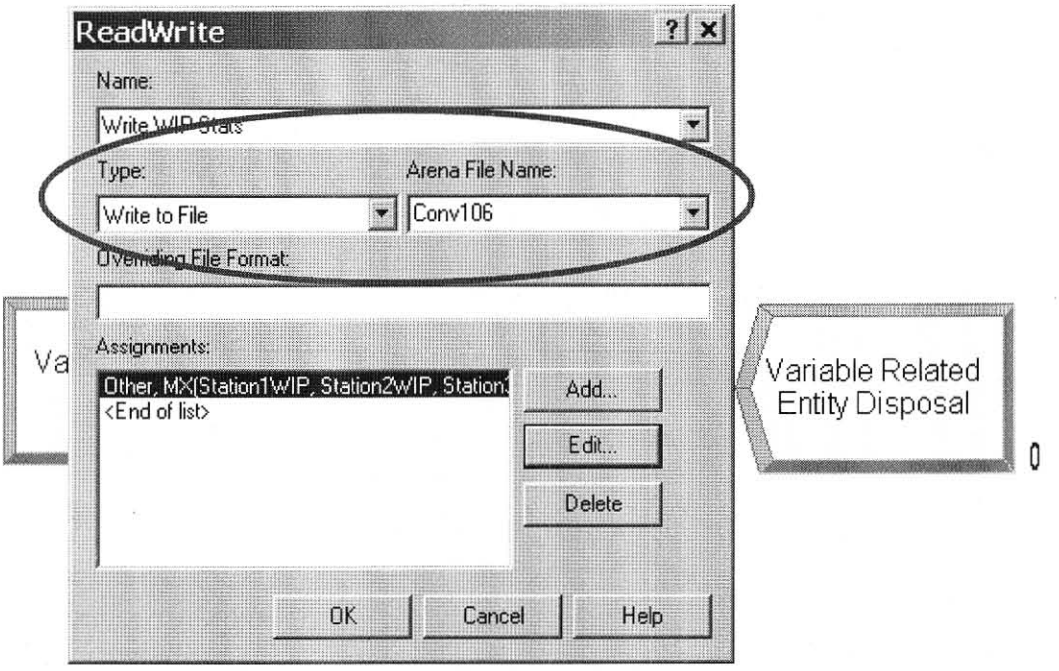


Figure 3-10: ReadWrite Module – Specifying Sequential Access File Name.

2. FILE STORAGE SPECIFICATION

In completing the ReadWrite module, the final step requires specification of a "save to" file location; as highlighted in Figure 3-11. Figure 3-12 represents an alternate method in which to specify file storage parameters for the experimental ReadWrite module. Both methods work in the same manner; file storage specifications are either made via "spreadsheet" (Figure 3-11), or via flow-model dialogues (Figure 3-12). Contingent upon these changes, Arena™ will record the maximum WIP value over all cells from each replication. Following this sequential file access, saved replication data is compiled and sent via TCP file transfer to the PRM application.

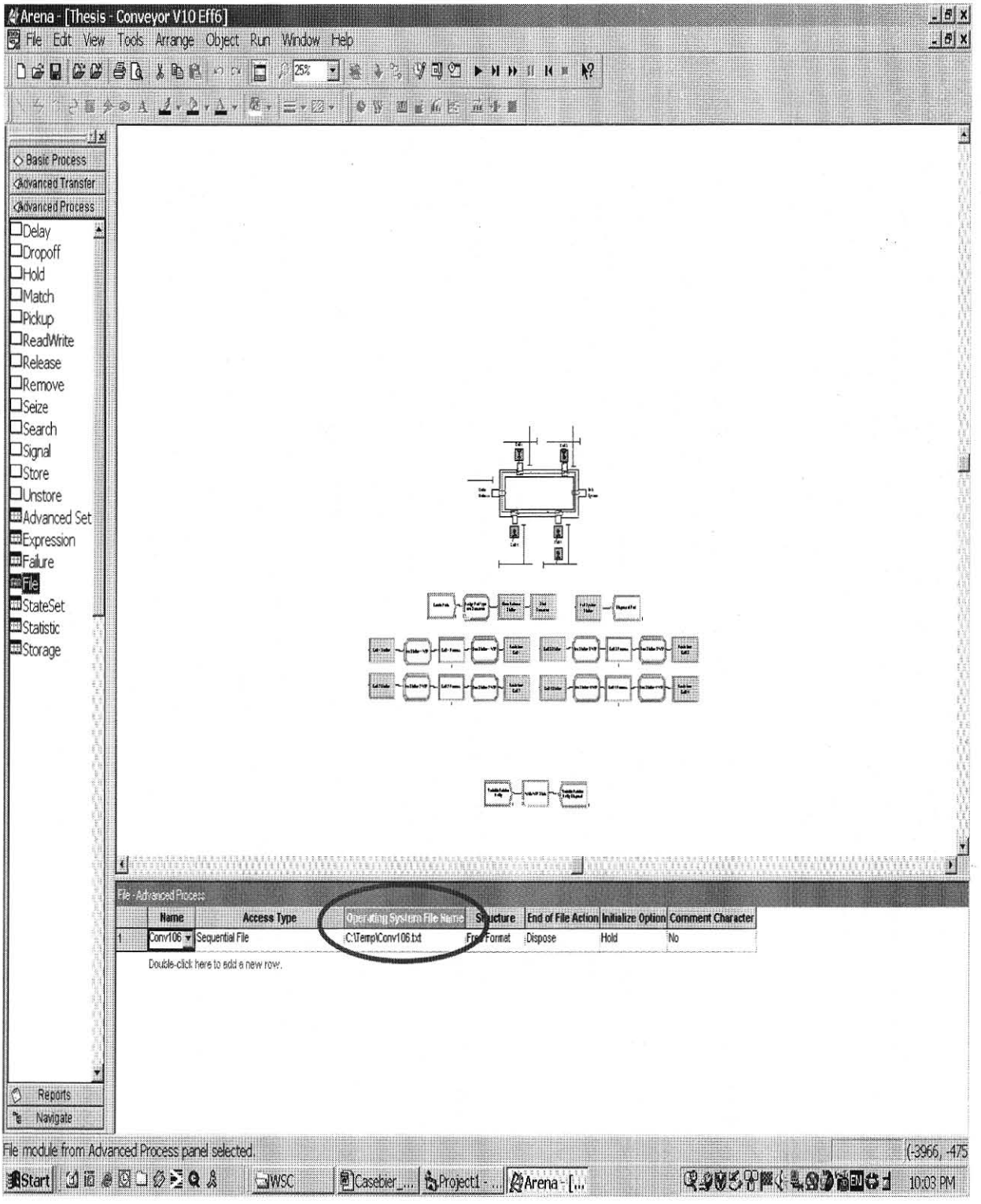


Figure 3-11: Advanced Process Module – File Storage Specification.

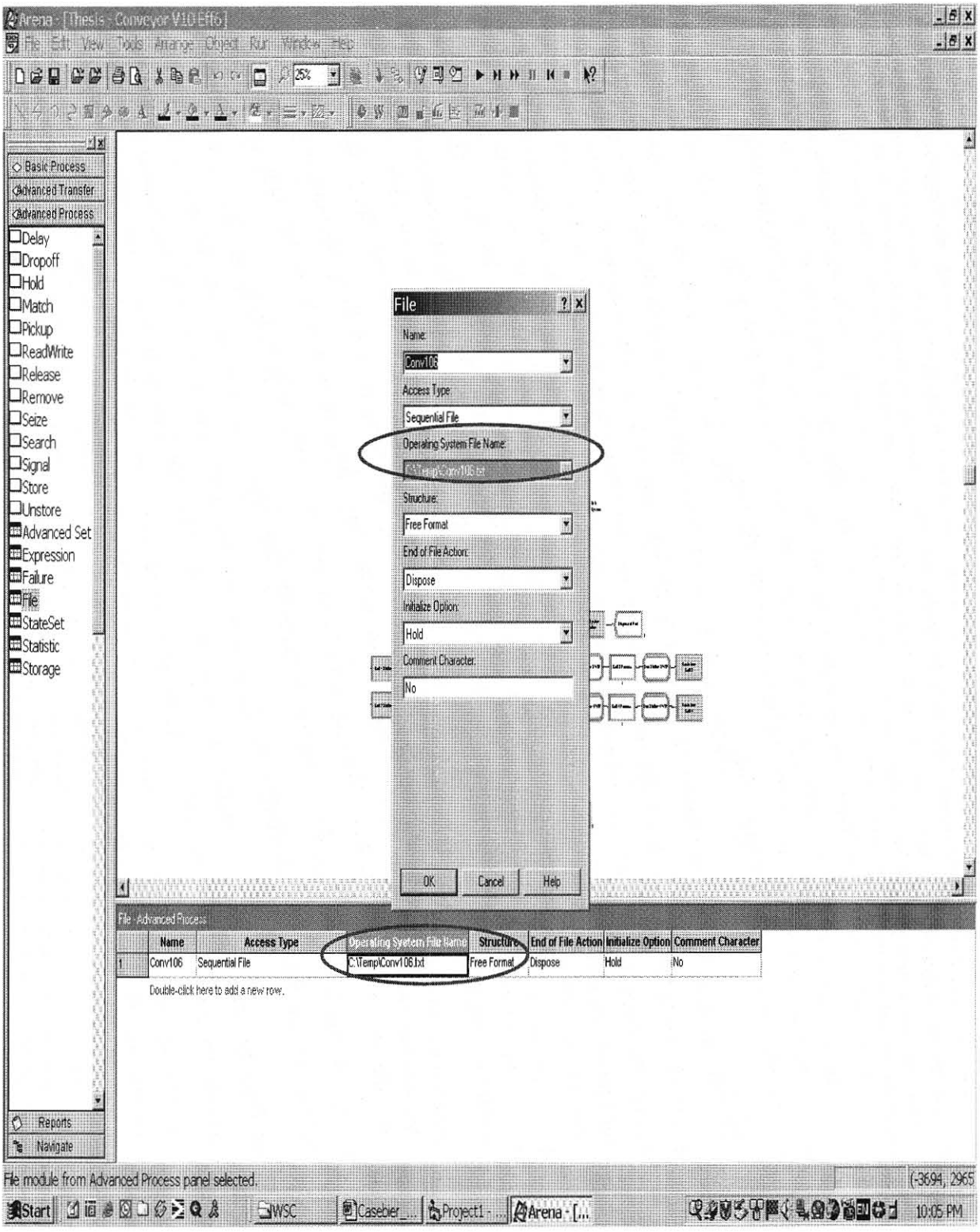


Figure 3-12: Advanced Process Module – File Storage Specification (Editing via Dialogue Box).

D. MODIFYING THE EXPERIMENTAL MODEL

Parameters requiring modification in the experimental model include, transport type, transport velocity, and cell three's efficiency factor. These variables constitute the three factors included in the 2^3 factorial design. Factor level modification may be passed as variables through the PRSE, however, this capability is not yet fully developed. Therefore, the experimental method employed in the present application utilizes multiple simulation models; each variation saved as its own unique model.

Models are either created on the "master" computer or by a remote analyst, they are transferred to "slave" computers via TCP file transfer (Appendix D). After receiving the simulation models, the "slave" computers are instructed to execute and compile the results of their respective models. This data is then sent back to the "master" for further analysis. Because of this method of factor manipulation, the method in which factors are changed need not be explained in great detail.

Transport types are differentiated by two separate experimental models (Models 8-2, and 8-4, Kelton, 2004). The efficiency factor (0.6 and 0.8, respectively) of work cell three is modified through the "variable" module located on the basic process template, and as described in Figure 3-13. Finally transport velocity is configured though the "conveyor" and "transport" modules found on the advanced transfer template, and as described in Figure 3-14. Through the modification of the aforementioned variables, and subsequent saving of respective models, the stage is set for the next phase of the experimental process.

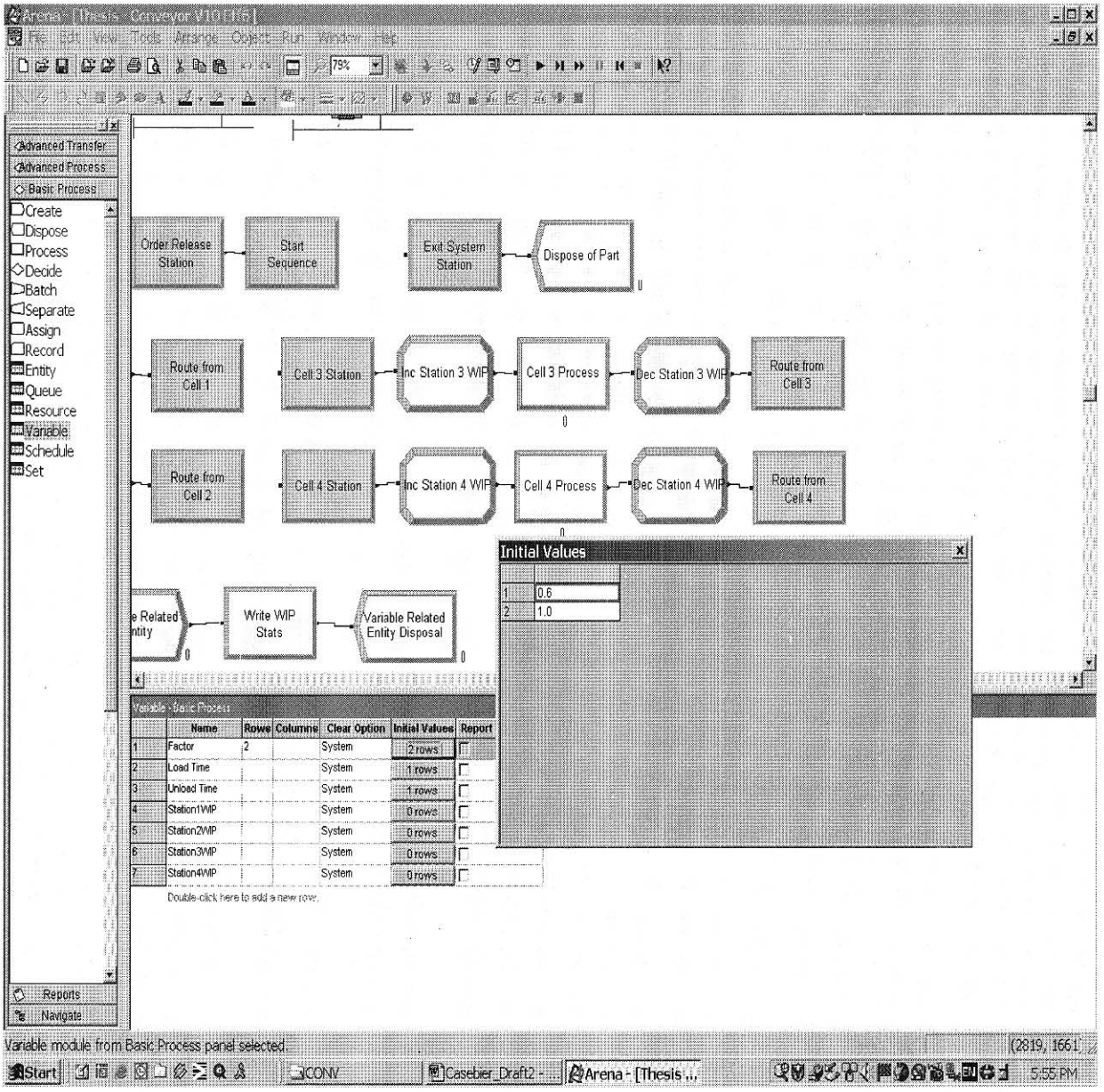


Figure 3-13: Configuring the Efficiency Factor of Work Cell 3.

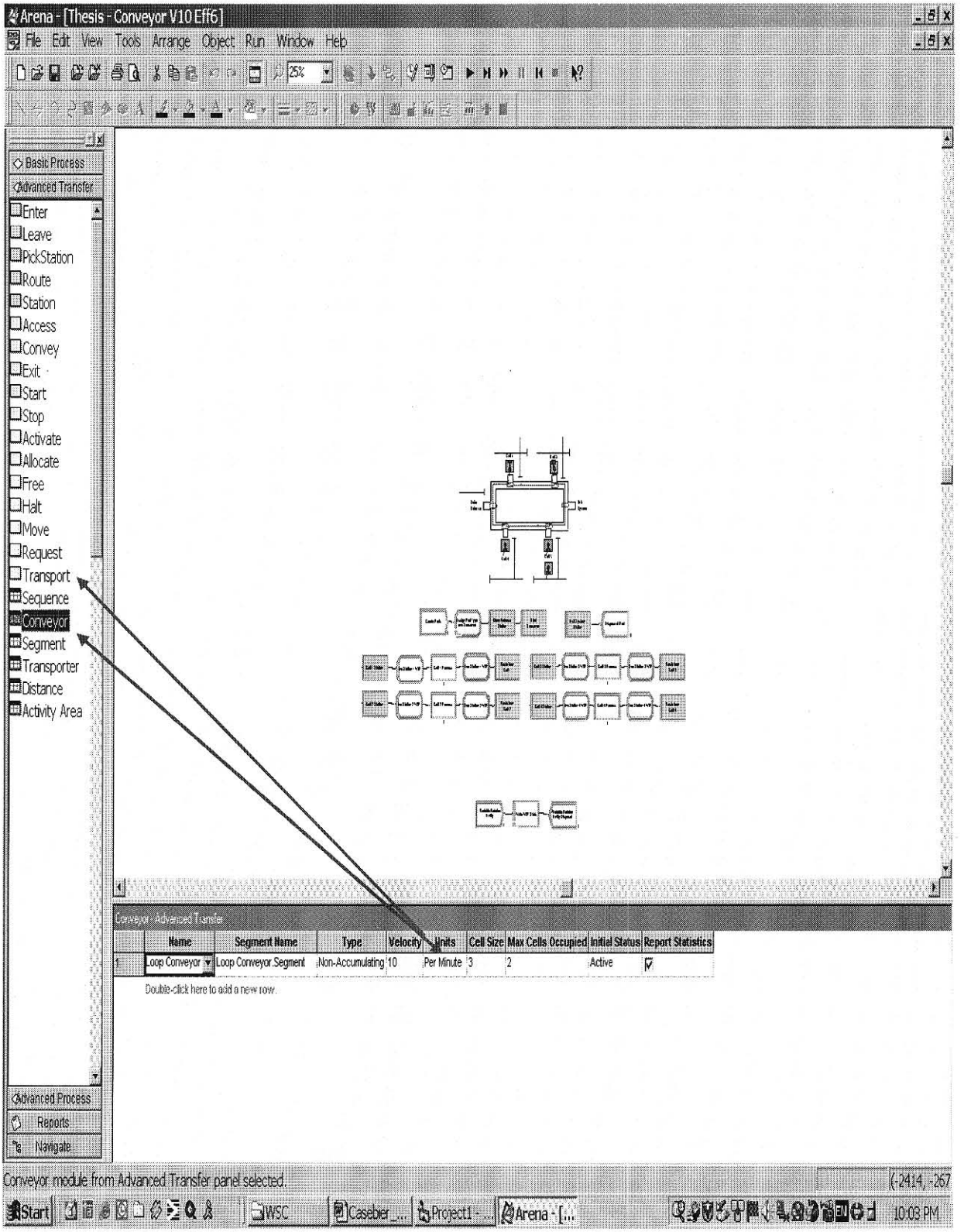


Figure 3-14: Conveyor/Transporter Velocity Modification.

CHAPTER IV

SOFTWARE DEVELOPMENT

A. SYSTEM OVERVIEW

The Parallel Replication Simulation System (PRSE) allows the analyst to build, run, and compile results of parallel replications executed on dedicated "slaves". This ability to enlist and execute parallel replications on "slave" computers is defined within the PRM application. This application utilizes two main components within Microsoft Visual Basic (VB) they are, Microsoft Common Dialogue Control, and Microsoft Winsock Control.

The common dialogue control is used to select simulation models to transfer to designated remote 'slaves' (Appendix D). Following model transmission, the PRC "remembers" the name of the last file transferred. At the discretion of the simulation analyst (PRM side), the aforementioned models are executed remotely (Appendix E) through the use of key words (Appendix F). This remote communication is done through the use of Transmission Control Protocol (TCP), which is part of the TCP/IP suite used to connect "hosts" over networks. The VB control allowing this type of communication is called the Winsock Control.

Among various types of communication connections, TCP was chosen over UDP (User Datagram Protocol) because of the connection type. TCP, unlike UDP, affords more error recovery services, thereby not only allowing the user to troubleshoot the

connection state, but TCP also ensures data delivery in the proper order. Figure 4-1 represents the functionality of this simulation configuration.

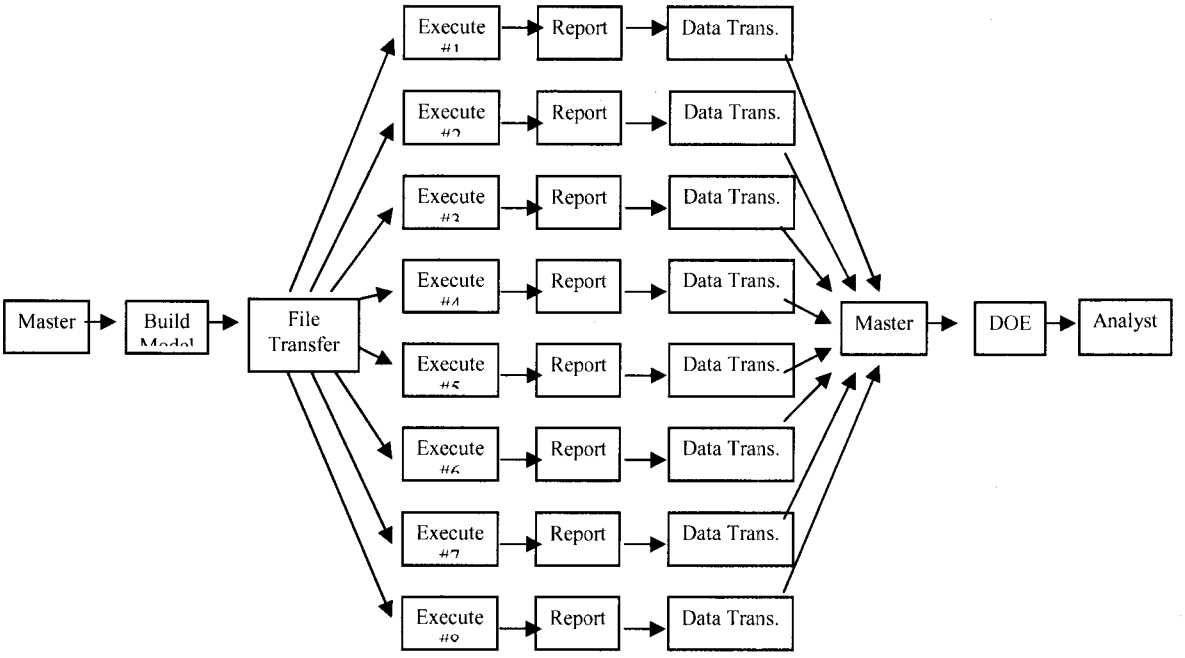


Figure 4-1: Functional block diagram for PRSS application and operation

B. ESTABLISHING A CONNECTION

Before the simulations can be remotely executed, a connection must first be established. This connection is established through a series of steps in which the "master" and "slave" begin to communicate. The following pseudo code demonstrates how this connection is initialized:

Client Application:

- Form_Load
  - tcpClient.LocalPort = 1001
  - tcpClient.Listen
  - lblConnectionStatus.Caption = "Listening... (not connected)"

The preceding pseudo code instructs the "Client", at form load, on what port to listen for instructions, and to display the current connection state. The following code instructs the "Master", also at form load, to prompt the user for a remote address, initialize time charts, what IP address to associate, and which port to make a connection with on the remote computer. The additional code initializes MSCharts used for displaying time savings data associated with executing parallel replications

Master Application:

- Form_Load
  - txtIP(n).text = InputBox(IPMessage, IPTitle)
  - chrt(n).ColumnCount = 8
  - chrt(n).RowCount = 1
  - tcpServer.RemoteHost = txtIP(n).text
  - tcpServer.RemotePort = 1001

C. SOLIDIFYING THE CONNECTION

Following this initial sequence of events, the "Master" and "Client" applications may begin the actual connection process. This process occurs in two steps, first the "Master" asks the "Client" to connect, secondly the "Client" replies to the "Master" acknowledging this request. This process can be exemplified through the use of a simple analogy, namely, making a phone call.

In order to make a phone call, the destination phone number (or IP address) must be known. Second, to speak to someone on the phone, the number must be dialed (tcpServer.Connect). Additionally, after the number is dialed, the person that is being called must hear the phone ring (listening) before it can be answered. Finally, if the person that is being called hears the phone ring and answers it, a conversation may begin. The following code demonstrates the connection request and solidification of the connection:

"Master" Application

- Private Sub tcpServer_ConnectionRequest(ByVal requestID As Long)
    - tcpServer.Connect
    - If tcpServer.State <> sckClosed Then
        - tcpServer.Close
    - End If
    - tcpServer.Accept requestID
- End Sub

"Client" Application

- Private Sub tcpClient_ConnectionRequest(ByVal requestID As Long)
    - If tcpClient.State <> sckClosed Then
        - tcpClient.Close
    - End If
    - tcpClient.Accept requestID
- End Sub

D. CHUNKING

In the preceding pseudo code, the connection is established between the "Master" and "Client" applications. Following this connection, the PRSE is now capable of sending models to "Clients", and invoking remote simulation replications. Model transmission is rather complex, however, the following pseudo code demonstrates this event. This example will assume that the user has already selected a file in which to transfer, however the actual code can be found in its entirety in Appendix D.

- Specify maximum data chunk size
    - On Event "Send File"
        - Save file name and path
            - Send to client
        - Wait for response
        - Send experimental model in increments of preset chunk size
            - Display transfer status

        - Wait for response
    - Display current connection state

A data chunk size must be specified in order to ensure accurate model delivery. The data chunk serves as a funnel, so to speak. This funnel ensures that each data chunk will fit into the "pipe", or Winsock Control, without truncation. From this, model integrity is maintained and the file transfer can occur with confidence.

E. EVENT SIGNALING

After the simulation models have been sent to their respective "slaves", the model can be executed. In order to remotely invoke an instance of Arena™ (Appendix E) and begin a parallel replication, an event identifier (Appendix F) must be passed from the "master" to "client" application. This event identifier is required because of the nature of the established connection. Although direct control of the remote computer may be possible through using Winsock Controls, the need for direct control is avoided by passing key words. The process that triggers the remote invocation of an Arena instance is given below.

"Client" Application

- If connection has been established Then
    - tcpClient.SendData "OK"
- End If

"Master" Application

- If txtPC(n).text = "Ok" Then
    - tcpServer.SendData "Go"
- End If

"Client" Application

- Private Sub txtSendData_Change()
    - If txtSendData.text = "Go" Then
        - Call OpenArena
    - End If
- End Sub

After the connection has been made, the "master" application receives notice that the connection is good ("OK"), and that it is possible to invoke a parallel replication. The parallel replication process is initiated by command button that the user clicks. This event triggers a key word, "Go", to be sent to the "client". After receiving the key word, the "client" is instructed to call a sub procedure, OpenArena, to invoke an instance of Arena™. Figures 4-2, and 4-3 represent user interfaces for both applications, however the PRC application requires no user interaction for functionality.

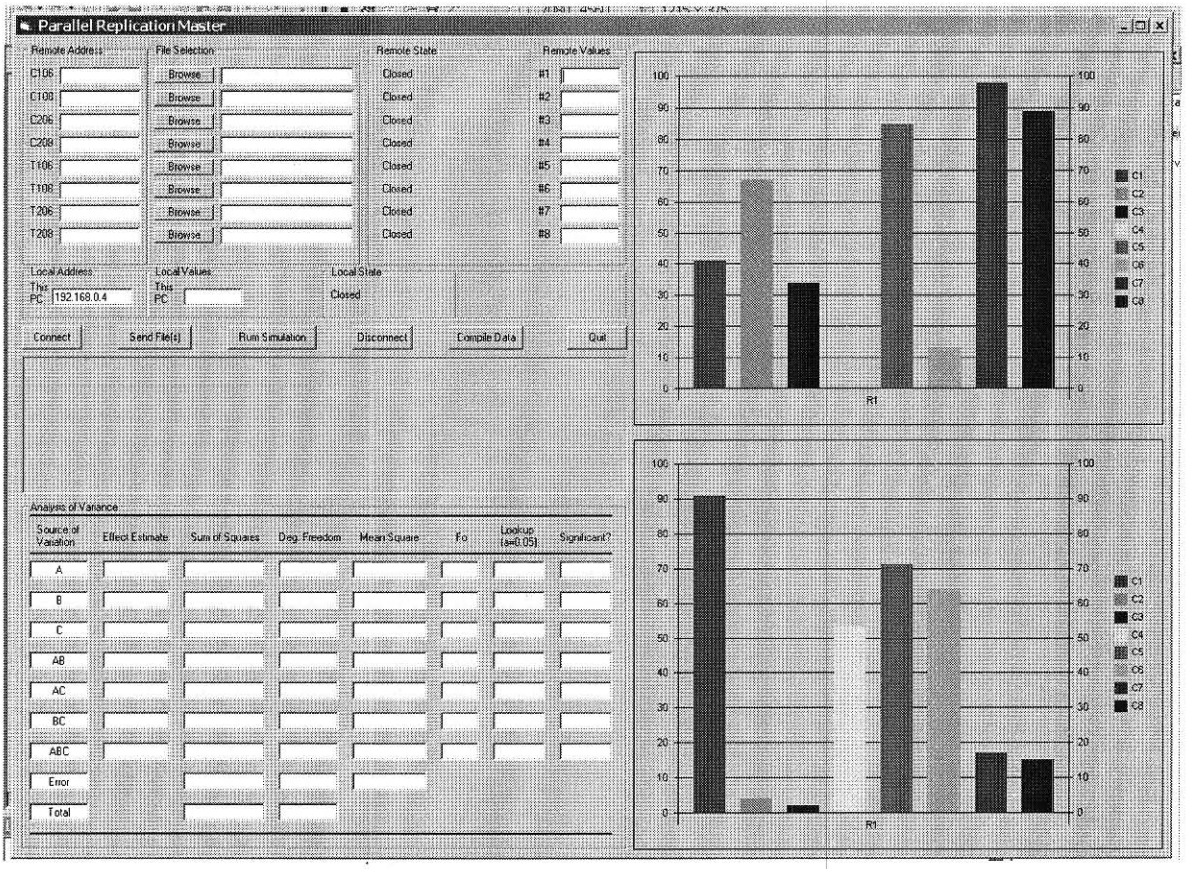


Figure 4 - 2: The PRM application

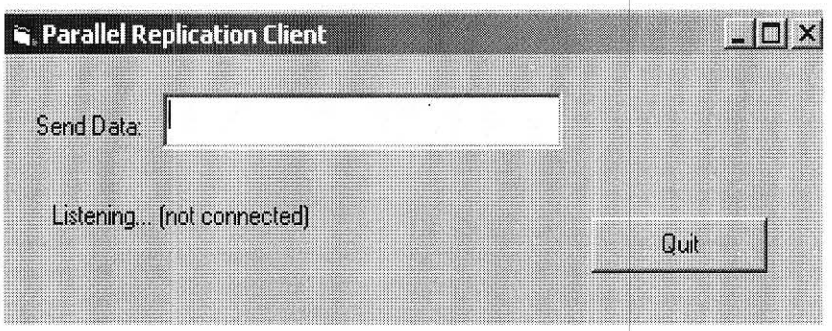


Figure 4 - 3: The PRC application

F. RECEIPT AND STORAGE OF EXPERIMENTAL DATA

Following the successful connection, model transmission, and remote simulation, the data must be transferred to the "master application" for analysis. This data transfer process occurs in similar fashion to the invocation of parallel replications (Appendix D). Again, a key word must be sent from the "client" to the "master". This process begins on the "client" side of the communication link after a simulation trial has been completed.

Following replication end, remote simulation data is placed into a "client" array. This data is then sent to the "master" application in the same manner in which key words are transferred (tcpClient.SendData). The "master" receives this data, reads it into an array, and then (following the end of experimental data transmission), writes it to a text file. In order for the "master" to know that it has received all data from a particular transmission, another key work must be passed in order to trigger the event. The key word identifying the completion of experimental data transfer is "Done".

CHAPTER V

THE EXPERIMENTAL DESIGN

A. OVERVIEW

Experiments are set up and performed in order to draw conclusions about a particular system. The method in which the experiments are set up and carried out is as important as the results obtained from the experiment itself. Modifying or fluctuating system variables one-at-a-time is often very costly and highly inefficient. Designed experiments allow the researcher to draw conclusions about a particular system in a more efficient and less costly manner.

In the present application, a 2^k factorial experiment is designed and executed across parallel processors. The results of these efforts produce a form of regression model known as the Metamodel (Kleijnen, 1979).

B. FACTOR LEVEL SELECTION

As previously mentioned, the simulation models used in this study primarily differ by the material handling equipment specified in each system. The response has been identified as work-in-progress, while the factors are transportation mode, velocity, and the efficiency factor associated with cell 3. Table 5-1 summarizes these experimental conditions. Each "slave" processor executes 30 replications for each factorial combination in order to obtain adequate statistical data (Appendix C).

TABLE 5-1: Variables associated with the Experimental Model

Response	Max. WIP
Factors (High)	Conveyor
	20
	Cell 3 efficiency – 80%
Factors (Low)	Transporters
	10
	Cell 3 efficiency – 60%

C. VARYING EXPERIMENTAL FACTORS

Parameters requiring modification in the experimental model include, transport type, transport velocity, and cell three's efficiency factor (old machine only, see Kelton, Sadowski, and Sturrock, 2004). These variables constitute the three factors included in the 2^3 factorial design. Factors may be modified by passing variables through the PRSE, however, this capability is not yet fully developed. Therefore, the experimental method employed in the current research utilizes multiple simulation models; each variation saved as its own unique model.

Whether these models are created locally, or by a remote analyst, models transferred to "slave" computers via TCP file transfer (as explained in chapter IV). After receiving the simulation models, the "slave" computers are instructed to execute and compile the results of their respective models. This data is then sent back to the "master"

for further analysis. Because of this form of factor manipulation, the method in which factors are changed need not be explained in great detail here, however future research will address this in its entirety.

The difference in transport mechanisms is addressed by the use of two separate experimental models (Models 8-2, and 8-4, Kelton, 2004). From these two initial models, the remaining experimental models may be obtained. The paragraph below summarizes how the remaining models are derived from manipulating the two remaining variables.

The efficiency factor (0.6 and 0.8, respectively) of work cell three is modified through the "variable" module located on the basic process template in Arena™, and as described in Figure 5-1. Second, transport velocity is configured though the "conveyor" and "transport" modules found on the advanced transfer template, and as described in Figure 5-2.

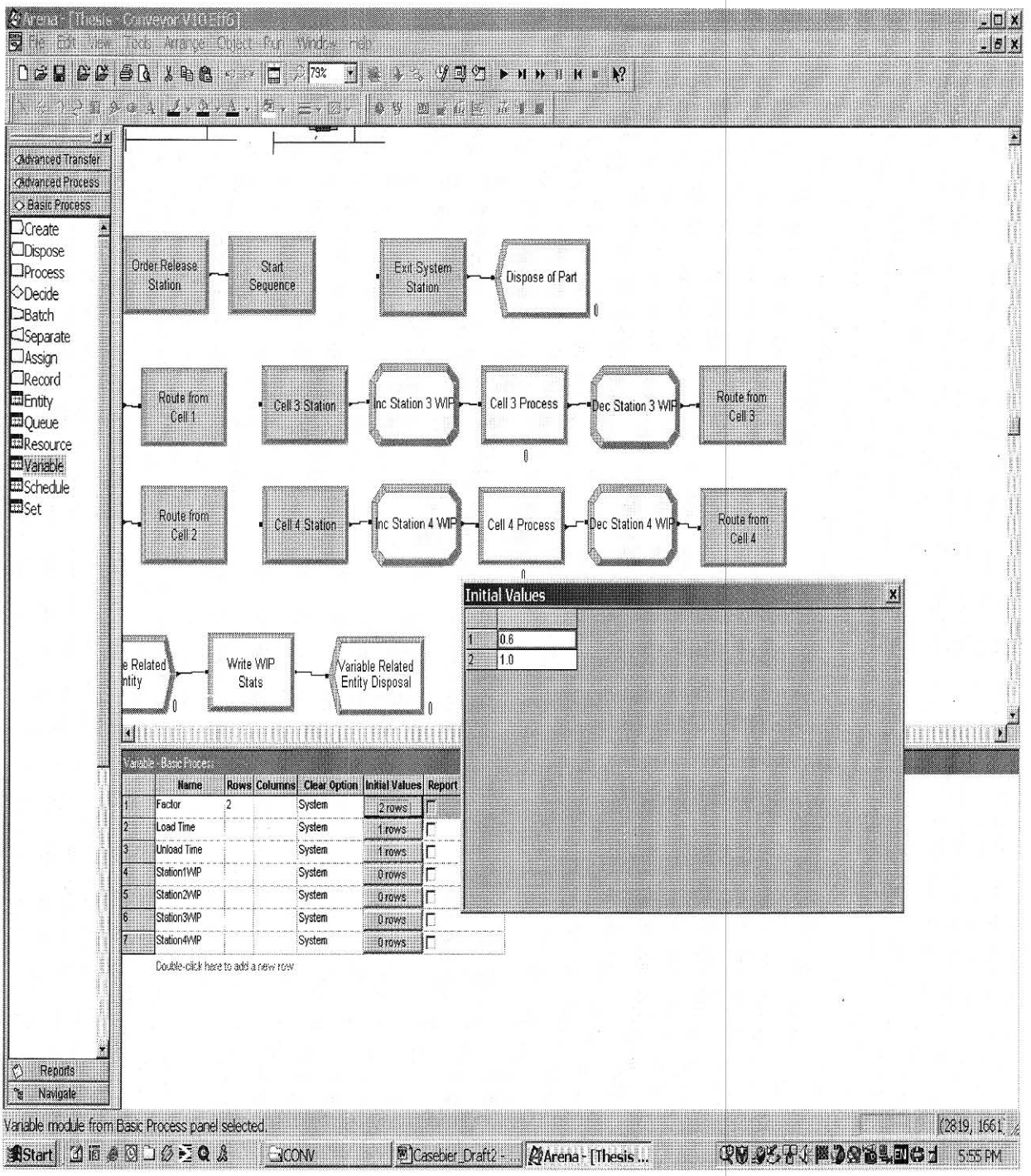


Figure 5-1: Configuring the Efficiency Factor of Work Cell 3.

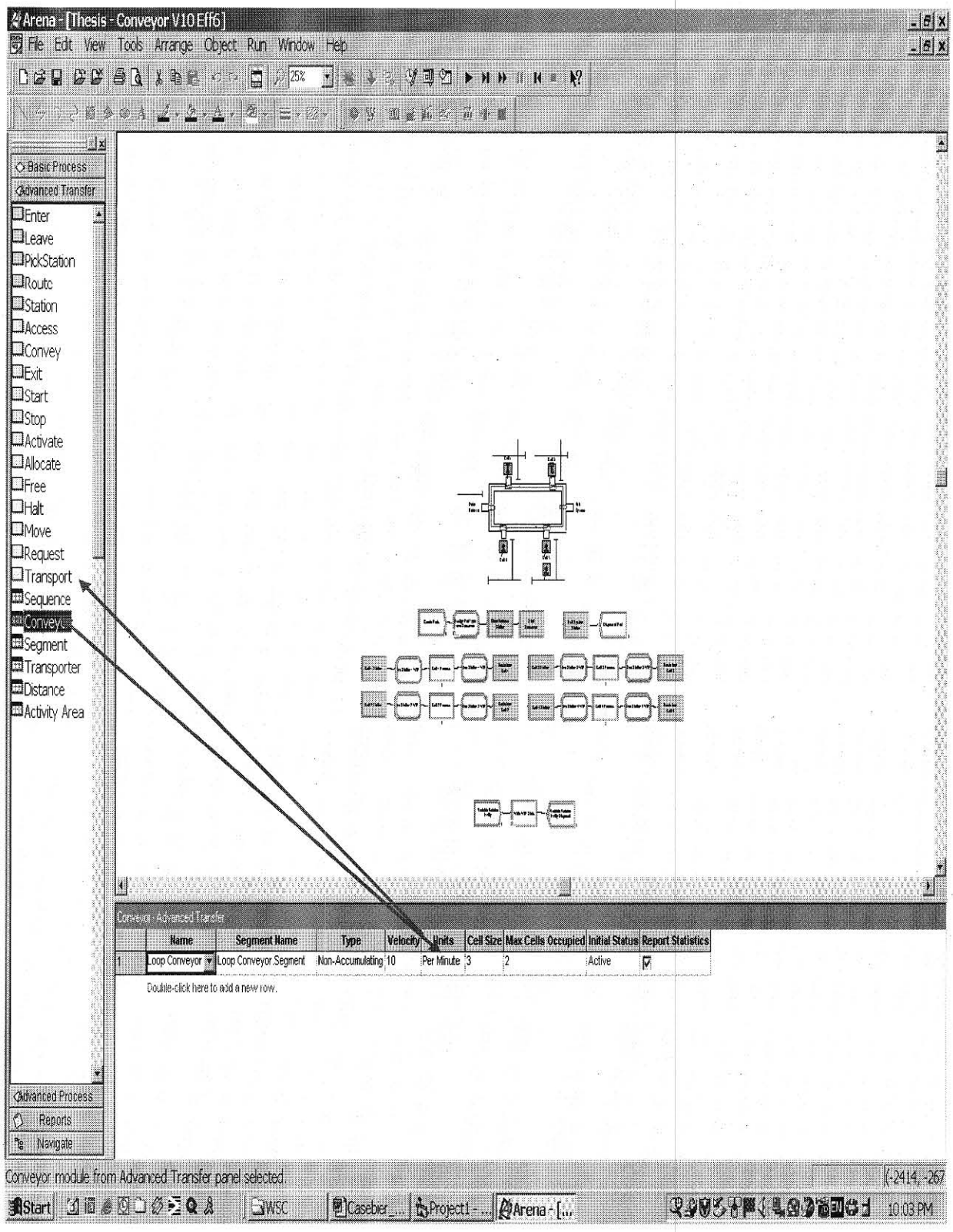


Figure 5-2: Conveyor/Transporter Velocity Modification.

D. THE FACTORIAL DESIGN

The designed experiment was performed from within the application via Microsoft Excel spreadsheet (Appendix C). The number of replications or data points in a factorial design corresponds to the number of factors in the experimental block. In this application, three factors are present thereby yielding $2^3 = 8$ replications.

Response values from the designed experiment can be found in Table 5 - 2. Randomization of experimental run order is negated because models are executed in parallel on different "slave" processors.

Table 5-2. Response Values from the Experimental Model.

Run Number	ABC Coding Scheme	Coding Interpretation	Response Value (Mean WIP)
1	- - -	C – 10 – 60%	3.83
2	- - +	C – 10 – 80%	4.13
3	- + -	C – 20 – 60%	4
4	- + +	C – 20 – 80%	4.17
5	+ - -	T – 10 – 60%	1.17
6	+ - +	T – 10 – 80%	1.2
7	+ + -	T – 20 – 60%	1.8
8	+ + +	T – 20 – 80%	2.2

TABLE 5-3: Associated factor coding levels.

Treatment Combinations	Design Factors								
	I	A	B	AB	C	AC	BC	ABC	Response
(1)	1	-1	-1	1	-1	1	1	-1	1.17
a	1	1	-1	-1	-1	-1	1	1	3.83
b	1	-1	1	-1	-1	1	-1	1	1.80
ab	1	1	1	1	-1	-1	-1	-1	4.00
c	1	-1	-1	1	1	-1	-1	1	1.20
ac	1	1	-1	-1	1	1	-1	-1	4.13
bc	1	-1	1	-1	1	-1	1	-1	2.20
abc	1	1	1	1	1	1	1	1	4.17

E. CALCULATIONS

Following the construction of the factorial experiment, calculations for the analysis of variance aid in determining the significance of system variables. The tabular values from the analysis of variance (ANOVA) are calculated in the following manner:

Sum of Squares Treatment,

$$SS_A = \sum_{i=1}^{a} \frac{y_{i\bullet\bullet\bullet}^2}{bcn} - \frac{y_{\bullet\bullet\bullet\bullet}^2}{abcn} \tag{4}$$

$$SS_B = \sum_{j=1}^{b} \frac{y_{\bullet j\bullet\bullet}^2}{acn} - \frac{y_{\bullet\bullet\bullet\bullet}^2}{abcn} \tag{5}$$

$$SS_C = \sum_{k=1}^{c} \frac{y_{\bullet\bullet k\bullet}^2}{abn} - \frac{y_{\bullet\bullet\bullet\bullet}^2}{abcn} \tag{6}$$

$$SS_{AB} = \sum_{i=1}^{a}\sum_{j=1}^{b}\frac{y_{ij\bullet\bullet}^2}{cn} - \frac{y^2{}_{\bullet\bullet\bullet\bullet}}{abcn} - SS_A - SS_B \tag{7}$$

$$SS_{AC} = \sum_{i=1}^{a}\sum_{k=1}^{c}\frac{y_{i\bullet k\bullet}^2}{bn} - \frac{y^2{}_{\bullet\bullet\bullet\bullet}}{abcn} - SS_A - SS_C \tag{8}$$

$$SS_{BC} = \sum_{j=1}^{b}\sum_{k=1}^{c}\frac{y_{\bullet jk\bullet}^2}{an} - \frac{y^2{}_{\bullet\bullet\bullet\bullet}}{abcn} - SS_B - SS_C \tag{9}$$

$$SS_{ABC} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}\frac{y_{ijk\bullet}^2}{n} - \frac{y^2{}_{\bullet\bullet\bullet\bullet}}{abcn} - SS_A - SS_B - SS_C \tag{10}$$

Sum of Squares Error (implicit),

$$SS_T = SS_{TREAT} + SS_E \tag{11}$$

Sum of Squares Total,

$$SS_{TOTAL} = \sum\sum\sum\sum y_{ijkl}^2 - \frac{y^2{}_{\bullet\bullet\bullet\bullet}}{abcn} \tag{12}$$

Degrees of freedom (*DF*) for each of the main effects and their interactions are given by the following formulae:

$$DF_A = a - 1 \tag{13}$$

$$DF_B = b - 1 \tag{14}$$

$$DF_C = c - 1 \tag{15}$$

Calculating degrees of freedom for model effects:

$$DF_{AB} = (a-1)\times(b-1) \tag{16}$$

$$DF_{AC} = (a-1)\times(c-1) \tag{17}$$

$$DF_{BC} = (b-1)\times(c-1) \tag{18}$$

$$DF_{ABC} = (a-1)\times(b-1)\times(c-1) \tag{19}$$

$$DF_{ERROR} = abc(n-1) \tag{20}$$

$$DF_{TOTAL} = abcn - 1 \tag{21}$$

Where,

a = the number of levels in factor A,

b = the number of levels in factor B,

c = the number of levels in factor C,

n = the number of blocks in the experimental model

The next step required in completing the ANOVA is to calculate the Mean Squares associated with each of the factors, and factor combinations.

$$MS_A = \left(\frac{SS_A}{DF_A} \right) \tag{22}$$

$$MS_B = \left(\frac{SS_B}{DF_B} \right) \tag{23}$$

$$MS_C = \left(\frac{SS_C}{DF_C} \right) \tag{24}$$

$$MS_{AB} = \left(\frac{SS_{AB}}{DF_{AB}} \right) \tag{25}$$

$$MS_{AC} = \left(\frac{SS_{AC}}{DF_{AC}} \right) \tag{26}$$

$$MS_{BC} = \left(\frac{SS_{BC}}{DF_{BC}} \right) \tag{27}$$

$$MS_{ABC} = \left(\frac{SS_{ABC}}{DF_{ABC}} \right) \tag{28}$$

$$MS_{ERROR} = \left(\frac{SS_{ERROR}}{DF_{ERROR}} \right) \tag{29}$$

An F-statistic is calculated in order to determine the significance of associated factors. The calculation of the statistic (in general), is as follows:

$$F_o = \left(\frac{MS_x}{DF_x} \right) \tag{30}$$

Where the subscript "*x*" represents the factor, or combination of factors in which the statistic is being calculated.

E. DATA ANALYSIS

Upon receipt of experimental data from remote PCUs, the data must then be exported to a spreadsheet for analysis (Appendices C, and I). Following the spreadsheet calculation, the data is then imported to the PRM application. To do this, the PRM application invokes a local instance of Microsoft Excel (Appendix H).

Table 5 - 4 presents the results of the analysis of variance used in determining whether factors or combinations of factors are significant. Accepting or rejecting the null hypothesis determines factor significance. Comparing calculated values, as shown in

equation 30, to a table look-up value, enables the analyst to determine whether the associated factor is or is not significant.

The response values obtained through executing parallel replications of the designed experiment represent the average maximum work-in-progress (WIP) observed from each experimental model. As previously noted, the purpose of this study is not to elaborate and prove the validity of experimental models, but rather to prove the effectiveness of carrying out distributed and parallel experiments over a network. With this being said, significant factors within the Metamodel can be identified from the analysis of variance (Table 5 - 4).

Table 5-4: Experimental Analysis of Variance (ANOVA)

Source of Variation	Effect Estimate	Sum of Squares	Degrees of Freedom	Mean Square	*fo*	Lookup (a = 0.05)	Significant
A	1.221	5.961736111	1	5.961736111	7.487348	5.32	**Yes**
B	0.229	0.210069444	1	0.210069444	0.263826	5.32	**No**
C	0.113	0.050625	1	0.050625	0.06358	5.32	**No**
AB	-0.179	0.128402778	1	0.128402778	0.161261	5.32	**No**
AC	0.004	6.94444E-05	1	6.94444E-05	8.72E-05	5.32	**No**
BC	0.029	0.003402778	1	0.003402778	0.004274	5.32	**No**
ABC	-0.063	0.015625	1	0.015625	0.019623	5.32	**No**
Error		6.36993	8	0.796241319			
Total		12.73986	15				

The analysis of variance showed that only one factor was significant in this experimental design, namely, transportation type. The response variable, MaxWIP, is minimized when using conveyors, and maximized when utilizing trucks (free path transporters). The resulting Meta-Model is displayed below.

The Resulting PRSE Meta-Model:

$$Y = \mu + \alpha_i\tau_i + \beta_i\tau_i + \gamma_i\tau_i + \alpha\beta_i\tau_i + \alpha\lambda_i\tau_i + \beta\lambda_i\tau_i + \alpha\beta\lambda_i\tau_i \quad (32)$$

Where, μ = grand average of responses over all parallel replications

α = effects estimate associated with factor A.

β = effects estimate associated with factor B.

γ = effects estimate associated with factor C.

τ = represents factor coding levels.

Because only transportation type was found to be significant within the experimental model, terms drop from the meta-model yielding equations 33 and 34.

$$Y = \mu + \alpha_i\tau_i \quad (33)$$

$$Y = 2.81 + (0.610417)\tau_i \quad (34)$$

The confidence level associated with this model is significantly low; therefore no additional discussion on the meta-model is necessary.

CHAPTER 6

CONCLUSIONS

A. OVERVIEW

The outcome of this study addressed the primary issue of executing remote and parallel simulation replications over a network array. Results were compiled and transferred back to the originating application, where statistical analysis and optimization took place. The present application will also time the execution of parallel simulations, however this functionality is not yet fully developed.

The software that has been developed for this purpose is in its infancy stages; however, the framework for future research in multiple languages has been set. Past research utilized the Java programming language, and classes contained therein. Java based software was developed, simulations conducted, and the results were analyzed. The current Microsoft-based application does something new, something that has not been done before, namely, utilizing turnkey software such as Arena™ in a network environment.

The PRM and PRC applications invoke instances of Arena™, perform remote simulation replications, and centrally compile the results for the local analyst. Although simple sounding, it has been said that the feat was "not possible". As this application is redefined and developed, additional capabilities and research possibilities will emerge.

The possibility of a distributed simulation API included with a commercially available software package is attractive to say the least.

As this system is streamlined, the objective will be to provide fast and reliable results for simulations with increasing complexity. The results will become available to the analyst in a timelier fashion. The ultimate goal will be to reduce the simulation time for complex models by a factor relating to the number of computers serving as 'slaves' in the array.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

As suggested, this study focused on researching the application of utilizing multiple computers to carry out a simulation study. This study employed a full-factorial designed experiment that was carried out over a network, and in some situations the world wide web. Ramifications of future study will drastically affect the time to carry out a full simulation study of complex modeling scenarios.

The PRM and PRC applications utilized only one of many available ports associated with a computers network card. Through the use of TCP/IP protocols, ports were utilized to communicate between various computers on a local area network. Additionally, other computers were used to carry out replications of a given modeling situation remotely, and with full capabilities. Given the opportunity, future research

studies investigating the maximum number of ports in which to run parallel replications would be extremely useful in determining capabilities of this application.

Another area of research, involves the determination of the number of ports that can be used to carry out multiple replications of a simulation over a given array of computers. Another possibility, how many simultaneous replications can be carried out over this array of ports on a network?

Additionally, another possible area of research could be the application of a large scale designed experiment, with thousands of factor levels. Traditionally, this type of experiment would have to be 'cut-down' through the use of various modeling techniques. The use of a fractional factorial design simplifies the analysis process, however this can only be effective on a certain level. This study showed that it is possible to distribute and run parallel replications of a simulation simultaneously across a network. Another avenue of interest would be to determine the amount of time required to drive a 2^{256} experiment (with full analysis) to completion.

Another possible area of future study would include the application of the PRM and PRC applications within the Professional Version of Arena™. The integration of this application with Optquest™ could prove to be extremely valuable, as optimization of simulation models within Arena™ is already possible.

REFERENCES

Anderson, N. P. 2004. "Simulation Optimization of Logistics Systems Through the Use of Criterion Models," Unpublished Ph.D. Dissertation, Department of Industrial Engineering, University of Louisville.

Biles, W. E., C. Marr, and C. Storey. 2000. A Java-Based Simulation Manager for Web-Based Simulation. In *Proceedings of the 2000 Winter Simulation Conference*, ed., J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1815-1822, The Society for Computer Simulation, San Diego, California.

Biles, W. E., Kleijnen, Jack, P. C. 1999. A Java-Based Simulation Manager for Optimization and Response Surface Methodology in Multiple-Response Parallel Simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 513-517, The Society for Computer Simulation, San Diego, California.

Biles, W. E, C. M. Daniels, and T. J. O'Donnell. 1985. Statistical Considerations in Simulation on a Network of Microcomputers. In *Proceedings of the 1985 Winter Simulation Conference,* ed., G. Gantz, G. Blais, S. Solomon, 388-393, The Society for Computer Simulation, San Diego, California.

Biles, W. E., G. W. Evans, Y. Khaskina and L. S. Cook. 1996. A best-of-k-systems approach to simulation with complex search. In *Mathematical Methods in Stochastic Simulation and Experimental Design*, S. M. Ermakov and V. B. Melas (Eds.), Publishing House of St.Petersburg University, 124-130.

Chen, C-H, I. Lee, Y-C. Luo, and E. Yucesan. 2000. Distributed Web-Based Simulation Optimization. In *Proceedings of the 2000 Winter Simulation Conference,* ed., J. A. Joines, R. R. Barton, K. Kang, and P. S. Fishwick, 1785-1793, The Society for Computer Simulation, San Diego, California.

Fujimoto, R. M. 1998. Parallel and Distributed Simulation. *Handbook of Simulation*, ed. J. Banks, 429-464.

Heidelberger, P. 1988. Discrete-event simulation and parallel replications: statistical properties. *Scientific and Statistical Computing* (9): 1114-1132.

Kelton, W. D., R. P. Sadowski and D. A. Sadowski. 1998. *Simulation with Arena*, McGraw-Hill, New York.

Kelton, D. W., R. P. Sadowski, and D. T. Sturrock. 2004. *Simulation with Arena, Third Edition,* McGraw Hill, New York, New York.

Kilgore, R., and K. Healy. 1999. Introduction to Silk: A Java-based, process-oriented simulation system. Threadtec, Inc., St. Louis, MO, 1-15.

APPENDIX A

Initializing a Connection

"PRM – Master"

```
Private Sub Form_Load()

'Call procedure to populate combo boxes with available IP addresses

'Determine IP addresses

        Dim IPTitle As String

        Dim IPDefault As String

        IPDefault = "192.168.0.3"

        IPTitle = "Enter the Remote IP Address"

        Dim IP1Message As String

IPMessage = "Enter Remote IP "


txtIP.Text = InputBox(IP1Message, IPTitle, IPDefault)


'The name of the Winsock controls are tcpServer1..8

'Note:  in general, to specify a remote host, you can use either

'      the IP address (ex: "121.111.1.1") or the computer's

'      "friendly" name (ex: "RemoteComputerName").


        tcpServer1.RemoteHost = txtIP1.Text

        tcpServer1.RemotePort = 1001

        Call ConnectionState

End Sub
```

"PRC – Client"

```
Private Sub Form_Load()
        'Make the PRC Application Invisible / Run in Background
         frmClient.Visible = True
        'Set the LocalPort proplerty to an integer
        'Then invoke the Listen method.
        tcpClient.LocalPort = 1001
        tcpClient.Listen
End Sub
```

APPENDIX B

Solidifying the Connection

"PRM – Master"

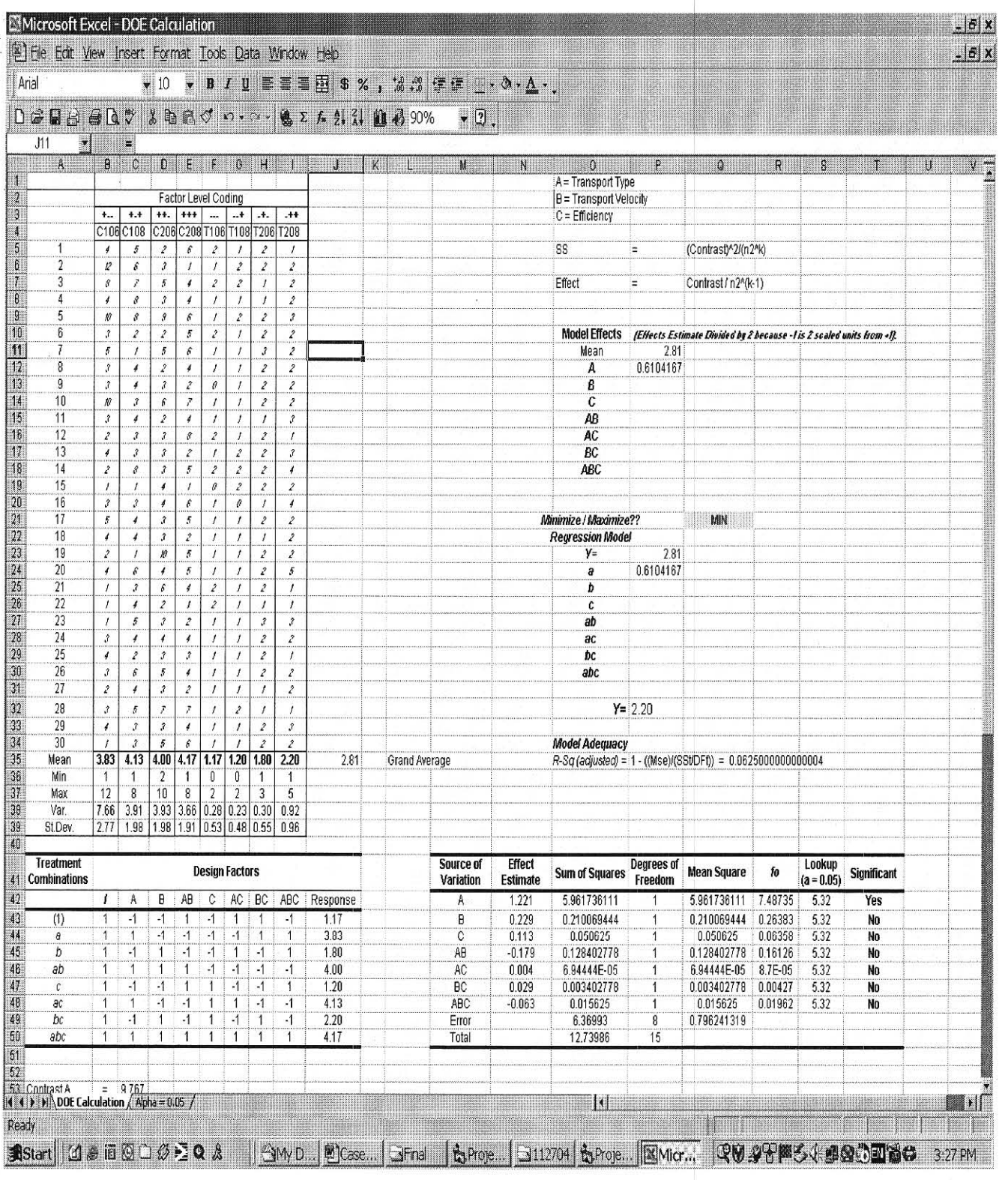
```
Private Sub cmdConnect_Click()
        On Error GoTo Canada:
        'Invoke the Connect method to initiate a connection.
        tcpServer1.Connect
        Call ConnectionState

    '   Open remote replication data files for replication data output.
        Open App.Path & "\" & "PC1Data.txt" For Output As #11
Canada:
End Sub

Private Sub tcpServer1_ConnectionRequest (ByVal requestID As Long)
    ' Check if the control's State is closed. If not,
    ' close the connection before accepting the new
    ' connection.
      If tcpServer1.State <> sckClosed Then _
      tcpServer1.Close
    ' Accept the request with the requestID parameter.
      tcpServer1.Accept requestID
End Sub
```

"PRC – Client"

```
Private Sub tcpClient_ConnectionRequest(ByVal requestID As Long)
    'Check if the control's state is closed.
    'If not, close the connection before accepting the new connection.
      If tcpClient.State <> sckClosed Then tcpClient.Close
    'Accept the request with the requestID parameter.
      tcpClient.Accept requested
      tcpClient.SendData "OK"
End Sub
```

APPENDIX C

Calculations with Microsoft Excel – A Screen Shot



Microsoft Excel - DOE Calculation

File Edit View Insert Format Tools Data Window Help

Arial 10

J11 =

	A	B	C	D	E	F	G	H	I	J
1										
2		Factor Level Coding								
3		+--	+-+	++-	+++	---	--+	-+-	-++	
4		C106	C108	C206	C208	T106	T108	T206	T208	
5	1	4	5	2	6	2	1	2	1	
6	2	12	6	3	1	1	2	2	2	
7	3	8	7	5	4	2	2	1	2	
8	4	4	8	3	4	1	1	1	2	
9	5	10	8	8	6	1	2	2	3	
10	6	3	2	2	5	2	1	2	2	
11	7	5	1	5	6	1	1	3	2	
12	8	3	4	2	4	1	1	2	2	
13	9	3	4	3	2	0	1	2	2	
14	10	10	3	6	7	1	1	2	2	
15	11	3	4	2	4	1	1	1	3	
16	12	2	3	3	8	2	1	2	1	
17	13	4	3	3	2	1	2	2	3	
18	14	2	8	3	5	2	2	2	4	
19	15	1	1	4	1	0	2	2	2	
20	16	3	3	4	6	1	0	1	4	
21	17	5	4	3	5	1	1	2	2	
22	18	4	4	3	2	1	1	1	2	
23	19	2	1	10	5	1	1	2	2	
24	20	4	6	4	5	1	1	2	5	
25	21	1	3	6	4	2	1	2	1	
26	22	1	4	2	1	2	1	1	1	
27	23	1	5	3	2	1	1	3	3	
28	24	3	4	4	4	1	1	2	2	
29	25	4	2	3	3	1	1	2	1	
30	26	3	6	5	4	1	1	2	2	
31	27	2	4	3	2	1	1	1	2	
32	28	3	5	7	7	1	2	1	1	
33	29	4	3	3	4	1	1	2	3	
34	30	1	3	5	6	1	1	2	2	
35	Mean	**3.83**	**4.13**	**4.00**	**4.17**	**1.17**	**1.20**	**1.80**	**2.20**	2.81
36	Min	1	1	2	1	0	0	1	1	
37	Max	12	8	10	8	2	2	3	5	
38	Var.	7.66	3.91	3.93	3.66	0.28	0.23	0.30	0.92	
39	St.Dev.	2.77	1.98	1.98	1.91	0.53	0.48	0.55	0.96	

Grand Average

A = Transport Type
B = Transport Velocity
C = Efficiency

SS = (Contrast)^2/(n2^k)

Effect = Contrast / n2^(k-1)

Model Effects *(Effects Estimate Divided by 2 because -1 is 2 scaled units from +1).*

Mean	2.81
A	0.6104167
B	
C	
AB	
AC	
BC	
ABC	

Minimize / Maximize?? MIN

Regression Model

Y=	2.81
a	0.6104167
b	
c	
ab	
ac	
bc	
abc	

Y= 2.20

Model Adequacy

R-Sq (adjusted) = 1 - ((Mse)/(SStDFt)) = 0.0625000000000004

Treatment Combinations	Design Factors								
	I	A	B	AB	C	AC	BC	ABC	Response
(1)	1	-1	-1	1	-1	1	1	-1	1.17
a	1	1	-1	-1	-1	-1	1	1	3.83
b	1	-1	1	-1	-1	1	-1	1	1.80
ab	1	1	1	1	-1	-1	-1	-1	4.00
c	1	-1	-1	1	1	-1	-1	1	1.20
ac	1	1	-1	-1	1	1	-1	-1	4.13
bc	1	-1	1	-1	1	-1	1	-1	2.20
abc	1	1	1	1	1	1	1	1	4.17

Source of Variation	Effect Estimate	Sum of Squares	Degrees of Freedom	Mean Square	*fo*	Lookup (a = 0.05)	Significant
A	1.221	5.961736111	1	5.961736111	7.48735	5.32	Yes
B	0.229	0.210069444	1	0.210069444	0.26383	5.32	No
C	0.113	0.050625	1	0.050625	0.06358	5.32	No
AB	-0.179	0.128402778	1	0.128402778	0.16126	5.32	No
AC	0.004	6.94444E-05	1	6.94444E-05	8.7E-05	5.32	No
BC	0.029	0.003402778	1	0.003402778	0.00427	5.32	No
ABC	-0.063	0.015625	1	0.015625	0.01962	5.32	No
Error		6.36993	8	0.796241319			
Total		12.73986	15				

Contrast A = 9.767

DOE Calculation | Alpha = 0.05

Ready

Start | My D... | Case... | Final | Proje... | 112704 | Proje... | Micr... | 3:27 PM

APPENDIX D

TCP File Transfer

```
'Determine data chunk size
Public Const MAX_CHUNK = 8192 'Max size of sendable data



"PRM – Master"
Private Sub cmdBrowse1_Click()
'Find the file that you want to transfer
cdOpen.ShowOpen
   If vbOK Then txtFileName1 = cdOpen.FileName
End Sub

Private Sub cmdSendFiles_Click()
   If txtFileName1 <> "" Then
      Pause 200
      Call Send1
   End If
   'Show the appropriate connection state
   Call ConnectionState
End Sub

   'Sends Files to Remote PC #1
```

```
    Dim Fname_Only As String
    If txtFileName1 = "" Then
        MsgBox "No file selected to send ...", vbCritical
    Else
    If tcpServer1.State <> sckClosed Then
        Fname_Only = GetFileName(txtFileName1)
        SendFile1 Fname_Only
    End If
    End If
End Sub


Function GetFileName(Fname As String) As String
    Dim i As Integer
    Dim TempStr As String
    For i = 1 To Len(Fname)
        TempStr = Right(Fname, i)
        If Left(TempStr, 1) = "\" Then
            GetFileName = Mid(TempStr, 2, Len(TempStr))
            Exit Function
        End If
    Next i
End Function
```

```
Sub SendFile1(Fname As String)
Dim DataChunk As String
Dim Passes As Long
SendData1 "OpenFile," & Fname
Pause 200
Open Fname For Binary As #1
   Do While Not EOF(1)
      Passes = Passes + 1
      DataChunk = Input(MAX_CHUNK, #1)
      SendData1 DataChunk
      frmServer.lblPC1.Caption = "Transferring ..." & (MAX_CHUNK * Passes) & " Bytes"
      Pause 200
      DoEvents
   Loop
SendData1 "CloseFile,"
"picTransferStatus.Print "Connected."
Close #1
Passes = 0
End Sub
```

"PRC – Client"

```
Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
   'Show that data is arriving
   lblStatus.Caption = "Receiving Data"

   Dim Command As String, NewArrival As String, Data As String
   Static DataCnt As Long
   Dim i As Integer

   'See what kind of transmission is coming over
   tcpClient.GetData NewArrival, vbString

   'Determine whether to run Arena or ....
   If NewArrival = "Go" Then
      txtSendData.Text = NewArrival
      Exit Sub
   End If

   'If NewArrival = "num" Then
   '   If SendDataCount <= numEntries Then
   '      tcpClient.SendData WIP(SendDataCount)
```

```
'       SendDataCount = SendDataCount + 1
'       NewArrival = ""
'       Exit Sub
'   End If
'   Exit Sub
'End If


'Determine whether to Save a File Transfer
If Len(NewArrival) < MAX_CHUNK Then
   For i = 0 To Len(NewArrival)
      If Mid(NewArrival, i + 1, 1) = "," Then
         Command = Left(NewArrival, i)
         Data = Right(NewArrival, Len(NewArrival) - (i + 1))
         Exit For
      End If
   Next
End If

Select Case Command
   Case "Accepted"
      bReplied = True
      lblStatus = "Connected"
```

```
Case "ServerClosed"
    Form_Load
    tcpClient.Close
Case "OpenFile"
    Dim Fname As String
    Fname = App.Path & "\" & Data
    Open Fname For Binary As #1
    lblStatus = "File opened .... " & Data
Case "CloseFile"
    Close #1
    lblStatus = "File Transfer Complete.... "
    Pause 1000
    lblStatus = "Listening.... (Connected)"
    DataCnt = 0
    numFilesTransferred = numFilesTransferred + 1
    Call Reconnect
Case Else
      'If NewArrival = "numnum" Then
      '   NewArrival = ""
      '   Exit Sub
      'End If

      'If NewArrival <> "" Then
```

```
            Put #1, , NewArrival
            DataCnt = DataCnt + 1
            lblStatus = "Recieving Data.... " & MAX_CHUNK * DataCnt & " Bytes"
        'End If
    End Select

    'Declare a variable for the incoming data.
    'Invoke the GetData method and set the
    'Text Property of a TextBox named txtSendData to the data.
    'Dim strData As String
    'tcpClient.GetData strData
    'txtSendData.Text = strData
End Sub
```

APPENDIX E

Invoking an Instance of Arena

"PRC – Client"

```
Public Sub OpenArena()

Dim sArenaFileName As String
Dim oAreanApp As Arena.Application
Dim oArena As Arena.Model
Dim oMod As Arena.Module

   Set oArenaApp = GetObject("", "Arena.Application")
      On Error Resume Next
         If Err.Number <> 0 Then
            Set oArenaApp = CreateObject("", "Arena.Application")
         End If
      Err.Clear

   sArenaFileName = app.Path & "\" & Fname
   sArenaFileNameWithoutDirectory = Right(sArenaFileName, Len(sArenaFileName) -
Dir(sArenaFileName) - 1)
   nArenaNum = oArenaApp.Models.Find(smFindName, Dir(sArenaFileName))
```

```
    If nArenaNum = 0 Then
        Set oArena = oArenaApp.Models.Open(sArenaFileName)
    Else
        Set oArena = oAreanApp.Models(nArenaNum)
    End If

  'Make Arena Visible?
  oArenaApp.Visible = False
  'Run the model
  oArena.Go
  'Close Arena
  oArena.Application.Quit
  oArena.Save
  'Show that Arena Data is being sent
  lblStatus.Caption = "Sending Arena Data"

End Sub
```

APPENDIX F

Using Key Words as Event Identifiers

"PRC – Client"

```
Private Sub tcpClient_ConnectionRequest(ByVal requestID As Long)
   On Error GoTo IDERROR
      'Check if the control's state is closed.
      'If not, close the connection before accepting the new connection.
      If tcpClient.State <> sckClosed Then tcpClient.Close
      'Accept the request with the requestID parameter.
      tcpClient.Accept requestID
      bInConnection = True
         'lblStatus = "Listening... Connected."
      'SendData "Accepted,"
      tcpClient.SendData "OK"
      'Show that connection has been established
      lblStatus.Caption = "Listening... (connected)"
   Exit Sub

IDERROR:
      MsgBox Err.Description, vbCritical
End Sub
```

"PRM – Master"

```
Private Sub cmdRunSim_Click()
    Dim Action As String
   Action = "Go"

   If txtpc1.Text = "OK" Then
      tcpServer1.SendData (Action)
      StartTime1 = Timer1
      txtpc1.Text = "Go"
   End If

   'Initializes timers
   StartTime1 = Timer1
   Call ConnectionState

End Sub
```

"PRC – Client"

```
Private Sub txtSendData_Change()
   'Clear values from txtSendData.text
   'If txtSendData.Text = "" Then
   '   GoTo Home:
   'End If

   'The TextBox control named txtSendData contains the data to be sent.
   'Whenever the value of the textbox changes to something other than
   '"Go", then the string is sent to the Server via the SendData method.

   If txtSendData.Text = "Go" Then
      'Show that Arena Simulation is being executed
      lblStatus.Caption = "Executing Arena Simulation"
      Call OpenArena
……Call GetArenaData
      tcpClient.SendData "Done"
   ElseIf txtSendData.Text <> "Go" And txtSendData.Text <> "" Then
      'Send value placed by Arena Calculation
      tcpClient.SendData txtSendData.Text
   'ElseIf txtSendData.Text = "OK" Then
```

```
    '   tcpClient.SendData OK
    End If

    If txtSendData.Text = "" Then
       Exit Sub
    End If

'Home:
End Sub
```

"PRM – Master"

```
Private Sub tcpServer1_DataArrival(ByVal bytesTotal As Long)
'Declare a variable for the incoming data.
'Invoke the GetData method and set the Text property of a TextBox named txtOutput to the data.
    ReDim PC1Data(1 To 30) As Single
    Dim strData1 As String

    tcpServer1.GetData strData1
    txtpc1.Text = strData1
```

```
'Determine when to start recording replication data
   If strData1 = "OK" Then
      txtpc1.Text = strData1
   End If

   If strData1 <> "OK" And strData1 <> "Done" Then

      PC1Data(Reps1) = strData1
         'The data is being written to a file that is opened during the connect event
         Print #11, PC1Data(Reps1)
      Reps1 = Reps1 + 1
   End If

   If strData1 = "Done" Then
      Call Close11
      'Call VBtoExcel
      EndTime1 = Timer1
      TTime1 = EndTime1 - StartTime1
      picDisplay.Print "Total PCU1 Running Time: "; 1000 * (TTime1); " milliseconds."
   End If

End Sub
```

APPENDIX G

Export Data to Excel

```
Public Sub VBtoExcel1()
   Open App.Path & "\" & "PC1Data.txt" For Input As #11
   Do While Not EOF(11)
      Input #11, dumvar
         numdata = numdata + 1
   Loop
Close #11

ReDim Data1(1 To numdata) As Single
   Open App.Path & "\" & "PC1Data.txt" For Input As #11
      For i = 1 To numdata
         Input #11, Data1(i)
      Next i
   Close #11

'Dimension variables and read in Replication data into an array

Dim appExcel1 As excel.Application
Dim wb1 As excel.workbook
Dim ws1 As excel.worksheet
```

```
'start excel (invisible)
Set appExcel1 = CreateObject("Excel.Application")
'open the workbook
Set wb1 = appExcel.WorkBooks.Open(App.Path & "\" & "DOE Calculation.xls")
Set ws1 = appExcel.ActiveSheet
Sheets("DOE Calculation").Select

'Write WIPData to DOE Calculation Workbook
Dim RG1 As Integer
RG1 = 5
  For i = 1 To numdata

    ws1.range("B" & RG1).Select
    ActiveCell.FormulaR1C1 = Data1(i)

    RG1 = RG1 + 1
  Next i

wb1.Save
wb1.Close
Set ws1 = Nothing
Set wb1 = Nothing
```

```
appExcel1.Quit

Set appExcel1 = Nothing



End Sub
```

APPENDIX H

Importing Data from Excel

```
Private Sub cmdCompile_Click()

'Take Data from Excel and Place in PRM Application

Dim appExcel As Excel.Application
Dim wb As Excel.workbook
Dim ws As Excel.worksheet

'Start excel (invisible)
Set appExcel = CreateObject("excel.application")
'open the workbook
Set wb = appExcel.WorkBooks.Open(App.Path & "\" & "DOE Calculation.xls")
Set ws = appExcel.ActiveSheet
Sheets("DOE Calculation").Select

'Write Read DOE Data from Calculation Workbook
Dim RG1 As Integer, RG2 As Integer, RG3 As Integer, RG4 As Integer, RG5 As Integer, RG6
As Integer, RG7 As Integer, RG8 As Integer, RG9 As Integer
RG1 = 42
RG2 = 43
```

```
RG3 = 44

RG4 = 45

RG5 = 46

RG6 = 47

RG7 = 48

RG8 = 49

RG9 = 50


    'Populate ANOVA Table

    'Effects Estimates

    ws.range("N" & RG1).Select

    txtEEA.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG2).Select

    txtEEB.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG3).Select

    txtEEC.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG4).Select

    txtEEAB.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG5).Select

    txtEEAC.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG6).Select

    txtEEBC.Text = ActiveCell.FormulaR1C1

    ws.range("N" & RG7).Select
```

```
txtEEABC.Text = ActiveCell.FormulaR1C1


'Sum of Squares
ws.range("O" & RG1).Select
txtSSA.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG2).Select
txtSSB.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG3).Select
txtSSC.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG4).Select
txtSSAB.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG5).Select
txtSSAC.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG6).Select
txtSSBC.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG7).Select
txtSSABC.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG8).Select
txtSSError.Text = ActiveCell.FormulaR1C1
ws.range("O" & RG9).Select
txtSSTotal.Text = ActiveCell.FormulaR1C1


'Degrees of Freedom
```

```
ws.range("P" & RG1).Select
txtDFA.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG2).Select
txtDFB.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG3).Select
txtDFC.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG4).Select
txtDFAB.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG5).Select
txtDFAC.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG6).Select
txtDFBC.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG7).Select
txtDFABC.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG8).Select
txtDFError.Text = ActiveCell.FormulaR1C1
ws.range("P" & RG9).Select
txtDFTotal.Text = ActiveCell.FormulaR1C1

'Mean Square
ws.range("Q" & RG1).Select
txtMSA.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG2).Select
```

```
txtMSB.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG3).Select
txtMSC.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG4).Select
txtMSAB.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG5).Select
txtMSAC.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG6).Select
txtMSBC.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG7).Select
txtMSABC.Text = ActiveCell.FormulaR1C1
ws.range("Q" & RG8).Select
txtMSError.Text = ActiveCell.FormulaR1C1

'Fo
ws.range("R" & RG1).Select
txtFoA.Text = ActiveCell.FormulaR1C1
ws.range("R" & RG2).Select
txtFoB.Text = ActiveCell.FormulaR1C1
ws.range("R" & RG3).Select
txtFoC.Text = ActiveCell.FormulaR1C1
ws.range("R" & RG4).Select
txtFoAB.Text = ActiveCell.FormulaR1C1
```

```
ws.range("R" & RG5).Select
txtFoAC.Text = ActiveCell.FormulaR1C1
ws.range("R" & RG6).Select
txtFoBC.Text = ActiveCell.FormulaR1C1
ws.range("R" & RG7).Select
txtFoABC.Text = ActiveCell.FormulaR1C1

'Lookup Values
ws.range("S" & RG1).Select
txtLookA.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG2).Select
txtLookB.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG3).Select
txtLookC.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG4).Select
txtLookAB.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG5).Select
txtLookAC.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG6).Select
txtLookBC.Text = ActiveCell.FormulaR1C1
ws.range("S" & RG7).Select
txtLookABC.Text = ActiveCell.FormulaR1C1
```

```
'Significant?
ws.range("T" & RG1).Select
txtSigA.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG2).Select
txtSigB.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG3).Select
txtSigC.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG4).Select
txtSigAB.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG5).Select
txtSigAC.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG6).Select
txtSigBC.Text = ActiveCell.FormulaR1C1
ws.range("T" & RG7).Select
txtSigABC.Text = ActiveCell.FormulaR1C1



'Display the results in the picture box
Dim GrandAvg As Single
ws.range("J" & 35).Select
GrandAvg = (ActiveCell.FormulaR1C1)
```

```
wb.Close

Set ws = Nothing

Set wb = Nothing

appExcel.Quit

Set appExcel = Nothing




Dim eqnFront As String

eqnFront = "Y = " & GrandAvg & " + "


If txtSigA = "Yes" Then

   eqnFront = eqnFront & txtEEA.Text & "(X1)"

End If


If txtSigB = "Yes" Then

   eqnFront = eqnFront & txtEEB.Text & "(X2)"

End If


If txtSigC = "Yes" Then

   eqnFront = eqnFront & txtEEC.Text & "(X3)"

End If
```

```
If txtSigAB = "Yes" Then
   eqnFront = eqnFront & txtEEAB.Text & "(X4)"
End If

If txtSigAC = "Yes" Then
   eqnFront = eqnFront & txtEEAC.Text & "(X5)"
End If

If txtSigBC = "Yes" Then
   eqnFront = eqnFront & txtEEBC.Text & "(X6)"
End If

If txtSigABC = "Yes" Then
   eqnFront = eqnFront & txtEEABC.Text & "(X7)"
End If

picDisplay.Cls
picDisplay.Print "The resultant Metamodel is .....  "
picDisplay.Print ""
picDisplay.Print eqnFront
picDisplay.Print
picDisplay.Print "Where, X(n) represents each factor or factor combination coding level."
```

```
End Sub
```

VITA

The author, John B. Casebier, is the son of Patricia L. Snow and the late John C. Casebier. Additionally, the author was born February 26, 1973 in Cedar Rapids, Iowa. Leaving Memphis, the author moved from his hometown to Louisville Kentucky, where he received a Bachelor's of Science in Industrial Engineering in August, 2003. This thesis will conclude his education at the J.B. Speed School of Engineering, however he is considering enrolling in a PhD program in the future.