2-1-2001

Self-organized critical & complex adaptive systems in a simulated manufacturing environment

Kaine Gill

ROCHESTER INSTITUTE OF TECHNOLOGY

SELF-ORGANIZED CRITICAL & COMPLEX ADAPTIVE SYSTEMS IN A SIMULATED MANUFACTURING ENVIRONMENT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN THE DEPARTMENT OF INDUSTRIAL AND MANUFACTURING ENGINEERING

COLLEGE OF ENGINEERING

BY

KAINE C. GILL

B.S. ROCHESETER INSTITUTE OF TECHONLOGY (2000)

ROCHESTER, NEW YORK

FEBRUARY 2001

DEPARTMENT OF INDUSTRIAL AND MANUFACTURING ENGINEERING

COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Kaine C. Gill
has been examined and approved by the
thesis committee as satisfactory for the
thesis requirement for the
Master of Science degree

Approved By:

Professor Paul H. Stiebitz, Thesis Advisor

Professor Madhu R. Nair

I, Kaine C. Gill, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 2/19/01 Signature of Author: _____

DEDICATION

This thesis is dedicated to my parents and family, without whom I never would have imagined this possible. Thank you especially to my parents who throughout my life have given me the latitude to explore my ambitions without rigid boundaries, providing subtle guidance and unwavering support and love.

ACKNOWLEDGEMENTS

I would like to acknowledge, with heart felt appreciation, the following people:

My thesis advisor Paul Stiebitz and fellow researcher Kevin Kelly, thank you for the exciting hours of concept development, and painful hours of debugging.

Madhu Nair, my thesis committee member, your comments have been extremely helpful in my presentation of material.

Anne Ostlund, my compatriot master's student and friend, you are always up for answering a simulation (or any other) question, no matter how elementary it is.

My parents, again, thank you for standing behind me, beside me, but never in my way.

My brother, you have been my guide and exemplar in so many of life's situations.

My sister, I hope I never disappoint you.

Thank you to my grandparents who I don't see nearly enough. You have offered me lessons in life and sanctuary in times of distress.

And, last but not least, thank you Amie for keeping me honest, sane, happy, and loved throughout the entire process.

I realize that I have forgotten so many people and I apologize. If you ever read this, then you probably know me well enough for me to say, "thank you too".

ABSTRACT

Gill, Kaine C. Self-Organized Critical & Complex Adaptive Systems in a Simulated Manufacturing Environment. (Under the direction of Paul H. Stiebitz)

The application of this thesis compares three different manufacturing theories, varying on resource allocation, to decide which performs best. Three Resource Allocation methods are analyzed including a static resource allocation method, a semi-dynamic resource allocation method, and a dynamic resource allocation method. The last methodology used in this thesis develops a resource allocation method intended to display complex behavior.

A Base Case manufacturing simulation is developed to represent a static resource allocation system. A Theory of Constraints manufacturing model using the Theory of Constraints process represents a semi-dynamic resource allocation system. A Complex Adaptive System manufacturing model using autonomous agents represents a dynamic resource allocation system.

The systems were analyzed at multiple stress levels for system to system performance, as well as for complex behavior. Dynamic resource allocation outperformed the semi-dynamic allocation and static allocation systems unilaterally. Some complex behavior was displayed for elements of some models, but as a whole, a trend of increasing complexity did not emerge as the models were analyzed from the Base Case, to Theory of Constraints, to the Complex Adaptive System.

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

THE CONSTRUCTIONIST COMPLEXITY THEORY

Complexity Theory (also known as Complexity Science) is an emerging field attracting researchers from myriad backgrounds; physics, cosmology, biology, computer science, anthropology, sociology, and economics are a few of the fields involved in research. Complexity Theory diverges from many other systems theories, taking a constructionist, or bottom up, view of systems. Most systems theories are reductionist, or top down in nature, taking large complicated systems and refining them into smaller more manageable pieces. Phillip Anderson, Noble Prize Laureate in physics comments: "The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe" (Waldrop 81).

Complexity Theory is constructionist because it builds systems from the ground up, allowing basic system agents or elements to interact with other agents using a finite set of rules results in complex self-organizing behavior (Waldrop 329). This emergent complex behavior is a missing element in a reductionist approach to system analysis. Essentially, Complexity Theory proves that the whole is greater than the sum of its parts.

MANUFACTURING AND COMPLEXITY THEORY

Manufacturing research into Complexity Theory has been minimal, but some related fields, such as Complex Adaptive Systems (CAS) Theory and agent-based architectures have had very strong involvement with manufacturing. Manufacturing research in Complex Adaptive Systems and Autonomous Agent Architectures is extensive. Many applications have addressed scheduling issues in manufacturing, especially the scheduling of products, machines and piece parts. Two examples of scheduling structures, the first by Lin and Solberg and the second by Ottaway and Burns, are discussed in detail in Chapter 2. These studies seek to improve flexible manufacturing environments such as job shops or other highly recursive and potentially highly automated processing environments (Lin and Solberg "Integrated Shop Floor Control Using Autonomous Agents"; Ottaway and Burns "Adaptive, Agile Approaches to Organizational Architecture utilizing Agent Technology"). There has been little agent-based research related to resource allocation in more traditional, highly manual, assembly line production.

PROBLEM STATEMENT

The objective of this research is to explore the performance of a manufacturing assembly line relative to both efficiency and Complexity Measures. In particular, the research is addressed via the following research questions:

1. Can an agent-based resource allocation scheme, applied to assembly line manufacturing, outperform a systemic scheme based on the Theory of Constraints and a static scheme with no resource allocation structure?

2. Can Self-Organized Criticality be observed in agent-based allocation schemes applied to assembly line manufacturing?

To answer the research questions above, three manufacturing simulations have been developed and evaluated. The manufacturing environment simulated is derived from Anne Ostlund's thesis on Virtual Manufacturing (Ostlund). The product, built on the ten work area assembly line, is a family of electric hand held sanders. All three simulation models of the sander line are built in ProModel ™, a simulation software tool. These models vary mainly in the way resources, or assemblers are allocated to do work in the systems.

The first scenario modeled is referred to as the Base Case (BC) simulation and it demonstrates a typical manufacturing line operating without dynamic resource allocation. The second scenario, referred to as the Theory of Constraints (TOC) system, uses static resource allocation in conjunction with dynamic allocation driven by the principles of The Theory of Constraints (Goldratt; Kelly 14-18). The final scenario uses principles from Complexity Theory, Complex Adaptive Systems (CAS) Theory, and Autonomous Agent Theories to dynamically alter resource allocation. This third scenario adheres to Pascale's four qualifiers of Complex Adaptive Systems described in Chapter 2.

Measurements of Success

System Measures have been identified to answer the first research question, while Complexity Measures are employed to answer the second research question. The System Measures include "Net Profit" (Kelly 16), Total Sales Dollars Produced, "Throughput" (Kelly 16), Total Late Fees, and Total Resource Cost. The Complexity Measures are used to determine whether the systems display Self-Organized Criticality, which is considered a good indicator of complexity (Axelrod and Cohen 104-105; Waldrop 304-309). These Complexity Measures are Late Fee Per Late Job, Percent of Actual Sales Captured, Time Between Late Exits, Number of Late Jobs Per Day, Throughput Per Day, and Maximum Daily Queue Length.

Thesis Outline

Chapter 2 contains a review of relevant literature pertaining to Complexity Theory and some of its key thinkers, the Theory of Constraints, Complex Adaptive Systems Theory, and Agent based Architectures as they pertain to a manufacturing environment.

Chapter 3 discusses in greater detail the three manufacturing theories to be employed by the ProModel ™ simulations, namely the Base Case, Theory of Constraints, and Complex Adaptive Systems models.

Chapter 4 details the construction of components common to each model for all three manufacturing methodologies, such as layouts, products produced, arrivals of products, System Measures, and Complexity Measures.

Chapter 5 identifies aspects of the Base Case Model that are unique, such as the model justification, resource allocation, special cases of the System Measures, and assumptions.

Chapter 6 identifies aspects of the Theory of Constraints model that are unique, such as the model justification, resource allocation, special cases of the System Measures, and assumptions.

Chapter 7 identifies aspects of the Complex Adaptive System model that are unique, such as the model justification, resource allocation, special cases of the System Measures, and assumptions.

Chapter 8 details the results of each model type (BC, TOC, and CAS) for all three model stress levels (Fast, Medium, and Slow Arrival Files), and their replications (if applicable).

Chapter 9 discusses the outcomes of the models with respect to the System Measures and Complexity Measures related to the Research Questions described earlier in the chapter.

Chapter 10 highlights potential avenues for future research.

CHAPTER 2 RELEVANT LITERATURE REVIEW

DIVERSE AREAS OF COMPLEXITY RESEARCH

Complexity and Complex Adaptive Systems Research have been applied in a wide range of areas. Fields with content as far apart in scale as cosmology and neural networks both use Complexity and Complex Adaptive Systems Theories to help gain a better understanding of how our universe and its components work. Other areas of research include some softer sciences such as sociology, anthropology, and economics (Bak "Self-Organized Criticality: A Holistic ..." 477-492; Waldrop 304-309). Applications of Complex Adaptive Systems Theory utilizing Agent based software systems are found in equally dispersed arenas, ranging from intelligent information infrastructures for Internet communication to intelligent highways.

Complex Adaptive Systems Theory has also had a strong following from the fields related to manufacturing, such as the optimization of supply chains, planning systems, scheduling systems, and reconfiguration of existing manufacturing structures into intelligent manufacturing systems (Shen and Norrie "Agent Based Systems for Intelligent Manufacturing: A State of the Art Survey").

CELLULAR AUTOMATA (CA)

Cellular Automata is a field of research, which studies the creation of emergent behaviors resulting from the interaction of the smallest system elements. Stanislas Ulam, an early researcher in CA, described Cellular Automata essentially as follows: Imagine a world where time is the ticking of a universal clock and space is a lattice of cells. Each cell contains a basic computer to determine its state at each time step of the clock. There are a finite set of states that any cell can be in and a finite set of rules that determine the cell's next state. At each tick of the clock the states of all the cells can change. The current state of a cell and the states of its bordering cells will determine the cell's future state (Waldrop 219). Cellular Automata are basic enough to be precisely defined, yet detailed enough to be thoroughly analyzed. In CA basic rules can elicit a magnificent array of program behaviors, most often represented graphically on a computer screen.

CLASSIFICATIONS OF COMPLEXITY

The root of study for CA in the past was to find what researchers call universality or a set of laws that describe when and how the rules of Cellular Automata can be applied to the real world (Waldrop 87). Steven Wolfram, recluse physicist and Mathematica software CEO and creator, developed a universality classification of Cellular Automata in 1984 during his undergraduate career at Cal-Tech. It was his assertion that all computer simulated Cellular Automata fall into one of four universality classes and that these classes were unique to CA. These four were identified in the following order (Waldrop 225-226).

I. Class I can be thought of as rather dull, essentially equivalent to a system with "doomsday" type rules. Within one or two time steps these systems settle into a predetermined state and never change.

II. Class II systems are slightly more interesting. After an initial flourish of randomized life on the screen, these systems quickly settle down into a static state that potentially has some oscillating elements.

III. Class III goes completely in the other direction displaying only chaotic behavior. A chaotic pattern never repeats itself and never settles down.

IV. Finally, Class IV systems are most interesting of all. These are the rarest systems, where just the right set of rules develop behaviors that do not yield stasis, but do not go to chaos either. Class IV systems generate endlessly changing CA that are governed by standardized rules yet behave in innovative and complex ways.

Half a decade after Wolfram's identification, Chris Langton, then a researcher at Los Alamos Center for Nonlinear Studies, examined the Wolfram Classification in light of his research on Artificial Life. He reordered the Classification to progresses from Class I (Dull Systems), to Class II (Semi-Interesting Systems), to Class IV (Complex Systems), to finally Class III (Chaotic Systems). He dubbed the term "Edge of Chaos" for the new placement of Class IV systems, capturing the idea that these systems can only exist under a tight range of parameters or conditions. If any of these parameters or conditions falter, the system will fall either into a Class II or Class III behavior becoming much less interesting or much more chaotic. Langton's view of the complexity

classification projected it beyond the confines of Cellular Automata into the areas of Dynamical Physics, Matter, and possibly Artificial Life (Waldrop 225-235). He saw the relationship as follows:

Cellular Automata

Classes I & II → "IV" → III

Where Cellular Automata systems progressed from very benign to somewhat interesting, to very interesting, then to chaotic (See **Figure 2-1**).

Dynamical Systems

Order → "Complexity" → Chaos

Orderly dynamical systems are equivalent to equilibrium systems and cyclical systems that are highly predictable. Chaotic systems toss and turn tumultuously forever; they are in no way predictable, and will never settle down. The complex systems exhibit behaviors of both predictable and chaotic tendencies (See **Figure 2-1**).

Matter

Solid → "Phase Transition" → Fluid

Crystalline solids' molecules (and non-crystalline to some degree) are structured in a very ordered, known, and stable fashion. During a "phase transition" this solid undergoes a change that occurs in a very finite volume, temperature, and pressure region. When it emerges as a fluid, the molecules slide over one another, and collide in a chaotic manner. This finite phase transition region dubbed the "boundary of chaos," "onset of

chaos," and ultimately the "edge of chaos" became a metaphor for the complexity movement. This is where matter behaves like both a solid and a liquid (See **Figure 2-1**).

Life

Too Static → "Life/ Intelligence" → Too Noisy

This is Langton's most hypothetical extrapolation, but ultimately the one that drove him on his life's path. He continued to study and research his Artificial Life (AL) at Los Alamos, and the Santa Fe Institute, and now is currently researching it independently and contracting his services (See **Figure 2-1**). Each one of these Class IV type systems are precariously perched between Class II stasis and Class III chaos.

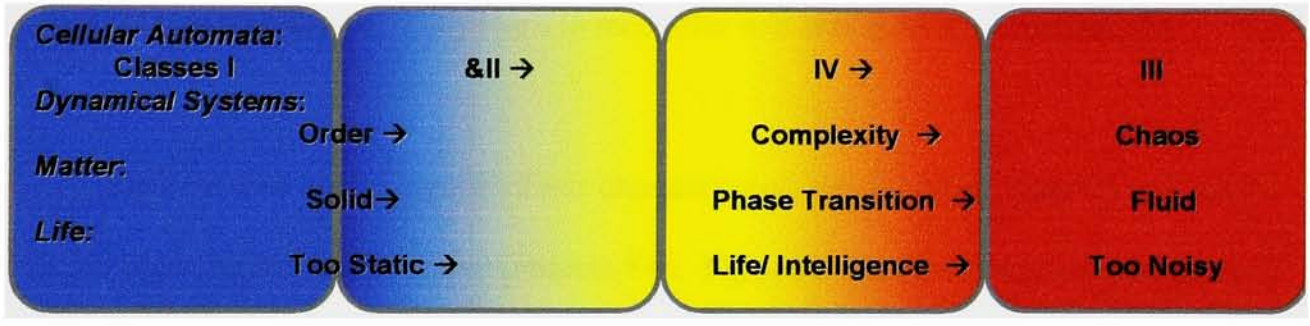


Figure 2-1 Langton's Interpretation of the Wolfram Classification

COMPLEX ADAPTIVE SYSTEMS (CAS)

Not only has complexity research begun a life of its own, but it also has spawned an offshoot. Complex Adaptive Systems research extends the concepts of complexity to various fields. Richard Pascale identifies four qualifiers for a Complex Adaptive System (84).

1. A CAS is comprised of many agents acting in parallel. Agents are the units of the system that interact with each other. They may be of one or many types and may or may not interact with all other agents. They must, however, interact with at least one other agent (Axelrod and Cohen 17). Control of the agents must be local, or non-hierarchical.
2. CAS agents are continuously shuffled to build multiple levels of organizational structure.
3. A CAS system must adhere to the second law of thermodynamics: entropy. It must continuously lose energy unless replenished by an outside source. This means Complex Adaptive Systems can die.
4. Complex Adaptive Systems exhibit a capacity for pattern recognition and employ this to anticipate the future. They learn to recognize and anticipate seasonal change.

Systems that obey some of these characteristics are complex, those that display them all are complex and adaptive (Pascale 84). **Figure 2-2** summarizes concepts drawn from Wolfram, Langton, and Pascale. Many of the qualities of CAS will be touched upon again in this chapter in the context of Agent Architectures.

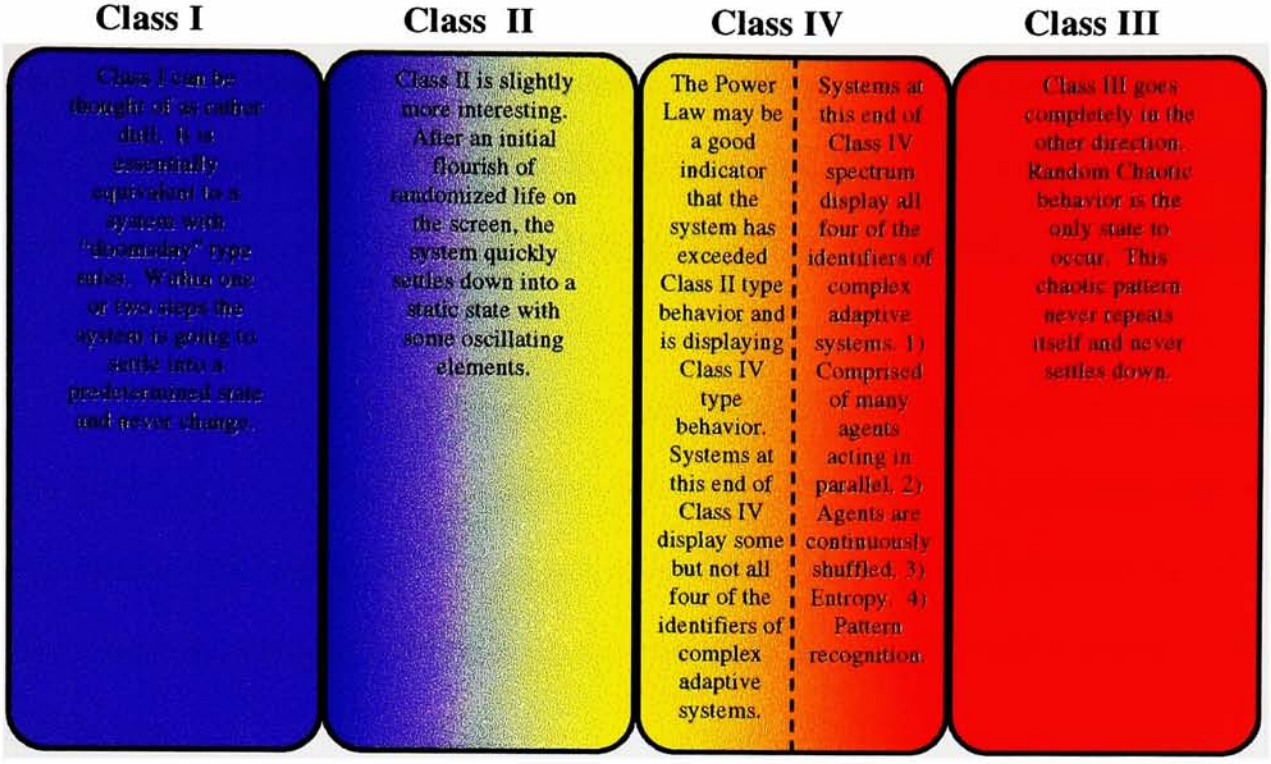


Figure 2-2 Modified Wolfram Complexity Classification

IDENTIFYING COMPLEX SYSTEMS

How does one identify a complex system? Per Bak, a physicist formerly at the Brookhaven National Laboratory, along with several other contributing researchers, has identified the concept of Self-Organized Criticality (SOC), which is considered one of the few analytical means of identifying complex systems (Axelrod and Cohen 104-105; Waldrop 304-306). SOC systems that drive themselves to a critical state where some of the behaviors that occur in the system fit a Power Law distribution.

Self-Organized Criticality is a phenomenon that, like Langton's extrapolation of the Wolfram Classes, seems to be applicable to many seemingly unrelated areas of research. Per Bak and his collaborating researchers developed the concept of Self-Organized Criticality around the same time that Langton was expanding the Wolfram Classes. Bak in essence describes SOC as the tendency of large-scale dynamical systems

to drive themselves to a critical state. A critical state in physics describes a system that is at a point of continuous phase transition (Bak "Self-Organized Criticality: A Holistic ..." 481). Continuous phase transition is analogous to the Class IV range of Wolfram's Cellular Automata, and Chris Langton's "Edge of Chaos" restructuring (See **Figure 2-1** and **Figure2-2**).

Many of the instances of Criticality that have been studied occur due to external intervention, i.e. a parameter of control such as magnetic field or temperature that an experimenter can manipulate. Bak has noticed that some critical behaviors occur in nature with no omnipotent external system driver, in essence they self-organize to this critical state.

Bak, along with fellow researchers, has observed potential SOC behavior in systems as diverse as economics, biology, seismology, cosmology, and anthropology (Bak "Self-Organized Criticality: A Holistic ..." 477-492; Waldrop 304-309). There is a universality that exists between all of these seemingly different fields. This universality is identifiable via the system behaviors or occurrences fitting a Power Law distribution. Power Law behaviors can be observed in actions occurring inversely proportional to a power of the magnitude or frequency of their occurrence (**Equation 2-1, Equation 2-2,** and **Equation 2-3**).

Equation 2-1: $P(s) \sim Ks^{-t}$, where s is the size of an occurrence, K is a constant, and t is the power of interest (Bak "Self-Organized Criticality in the 'Game of Life'" 342). Levels of t that display Power Law behavior have been frequently observed between $1[t [2$ for some scale free phenomena (Bak "Complexity, Contingency, and Criticality" 6693).

Equation 2-2: $P(S=s) \sim s^{-(1+b)}$, where s is the magnitude of an occurrence, (1+b) is the power of interest (Bak "Self-Organized Criticality: A Holistic View of Nature" 483).

Equation 2-3: $S(f) \approx 1/f^{b}$, where f is the frequency of an occurrence, and b is the power of interest. This phenomenon is also known as 1/f, flicker noise, and pink noise (Bak, Tang, and Wiesenfeld "Self-Organized Criticality" 373).

Self-Organized Criticality is typical of scale free systems as diverse as earthquakes, species extinction, and stock market fluctuations, where there is no typical size, magnitude, or frequency of occurrence. As indicated, these behaviors can also be scale free on spatial and temporal measures, in addition to magnitude. For example, the extinction of species can happen over disperse geographical areas or in centralized areas, and they can occur very rapidly in succession or over a long span of time (Bak "Self-Organized Criticality: A Holistic..." 479-480). The Power Law is a suggested empirical measure identified that is able to distinguish systems on the edge of chaos (Waldrop 327).

SELF-ORGANIZED CRITICALITY - THE SAND PILE METAPHOR

The sand pile is Bak's most famous SOC analogy. Imagine an empty table with grains of sand being dropped on it one at a time. At first nothing exciting happens, the grains just fall and stay more or less where they land. This is analogous to a Class I Wolfram state. Soon the sand forms a pile that becomes steeper and steeper leading to small avalanches or sand slides. This is comparable to a Class II Wolfram system. Finally, a point is reached where the amount of sand added is statistically equal to the amount of sand that falls off the edges of the table. This is where the system becomes interesting, and in Bak's terminology, Self-Organized and Critical. Avalanches in a SOC state range in size from one grain to the collapse of the entire pile. The size of the largest sand slide observed is proportional to the length of time one watches the system. The system has been loaded over history and is now acting as one complex system, as opposed to individual grains of sand acting dynamically and independently (Bak "Complexity..." 6691).

Figure 2-3 is a plot of avalanches with the size of each avalanche (s) on the x-axis, and the probability of an avalanche of that size P(s) on the y-axis. A sand pile at a SOC state is likely to have an avalanche of various magnitudes at frequencies inversely proportional to some power of the magnitude (Bak "Complexity, Contingency, and Criticality" 6691). When plotted on a log - log scale, a linear relationship is shown, which is indicative of systems displaying SOC behavior. A linear fit in log-log scale indicates that the system behavior fits a Power Law equation. These Power Law fits are used as indicators of complexity in an effort to answer research question two.

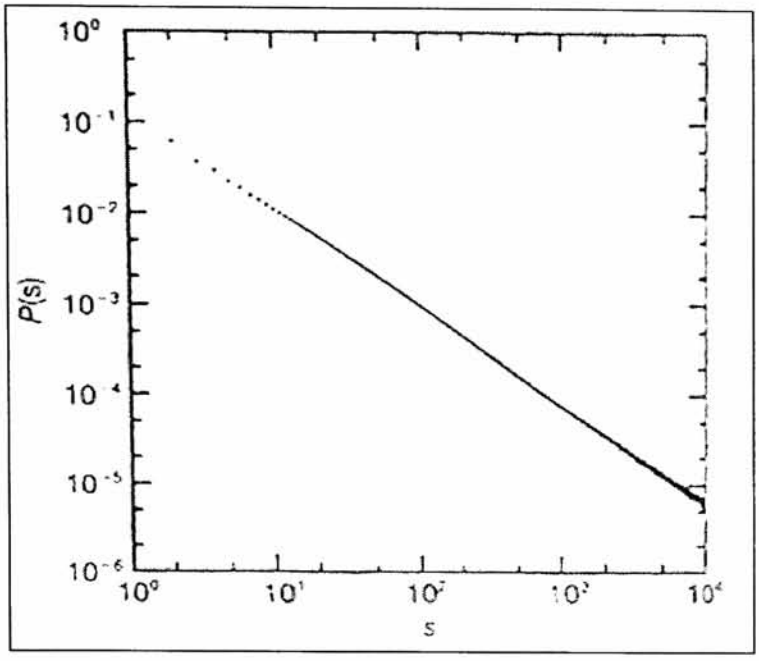


Figure 2-3 Per Bak's Sand Pile Avalanche Power Law Plot

THEORY OF CONSTRAINTS:

The Theory of Constraints (TOC) is an ongoing process improvement methodology developed by Eliyahu M. Goldratt. It is Goldratt's contention that the purpose of any company is to make money. Goldratt calls this The Goal, which is also the name of his most famous book.

MEASURES OF TOC

For the TOC process to be implemented, a measure is required that can identify constraints. Often in manufacturing and in this thesis the measure used is utilization. Goldratt often identifies constraints using utilization among other measures. Measures of progress in Goldratt's manufacturing methodology include "Throughput", "Inventory", and "Operating Expenses" (Goldratt 297). Goldratt defines Throughput as the sales

revenue generated for products built to a demand, less the cost of materials (Kelly 16). This definition is different than what is often used in manufacturing, where throughput represents the number of items produced, whether there is a demand or not.

Inventory is any money spent on purchasing things the company intends to sell. Operating Expense is any expense related to the cost of turning Inventory into Throughput. According to Goldratt, these measures are related in the following way (**Equations 2-4, 2-5**; Kelly 16):

Equation 2-4: Net Profit = Throughput - Operating Expense

Equation 2-5: Return on Investment = Throughput/Inventory

THE THEORY OF CONSTRAINTS PROCESS

The TOC leads ultimately to the following 5 focusing steps (Goldrattt 301; 307 and Kelly 18).

1. **Identify the system's constraint(s).** This step refers to finding the bottleneck(s) in a process. Bottlenecks, or constraints, may be physical constraints such as machine capacities and labor capacities, or policy constraints, such as batch size.
2. **Decide how to exploit the system's constraint(s).** Exploiting a constraint assures that it is utilized at the highest rate possible. The constraint should only work on product that becomes Throughput. Product that is built on the constraint that goes to stock is wasted time on the constraint. Likewise, a time buffer should be placed in front of the constraint so that it is never starved for work. A time buffer, unlike a traditional buffer, represents an amount of processing time at a bottleneck, not a set number of parts. Protective capacity at workstations that supply the time buffer in front of the constraint is required to keep the constraint constantly fed with work. Maximization of uptime of the constraint minimizes time lost in the system. If the constraint has nothing to work on, it will not be able to produce, and that time will be lost forever.
3. **Subordinate everything to bottleneck.** All other work center schedules are driven by the constraint. The completion of work at the constraint triggers upstream production. The "Drum Buffer Rope" (DBR) product release mechanism described later in this chapter is the most often-used subordination process.

4. **Elevate the system's constraint(s).** This step must be done fourth, and not first. Elevation can be accomplished by adding more capacity, as in new equipment, additional shifts, and/or capital upgrades.
5. **WARNING!** If in the previous steps a constraint has been broken, go back to step one, but do not allow inertia to cause a system constraint. In The Goal Goldratt gives the following example of inertia: a materials manager decided to build product to stock in order to keep the current bottlenecks producing at high utilization, even though there was no demand for the product (Goldratt 303-308). If by elevating the constraint another work center or process becomes the constraint, the TOC process starts over again at step one. Resisting inertia refers to adhering to the TOC process and not letting the current course of action influence future actions.

DRUM BUFFER ROPE

Extending the five focusing steps to execution most often is accomplished via the Drum Buffer Rope production control system, as illustrated in **Figure 2-4.**

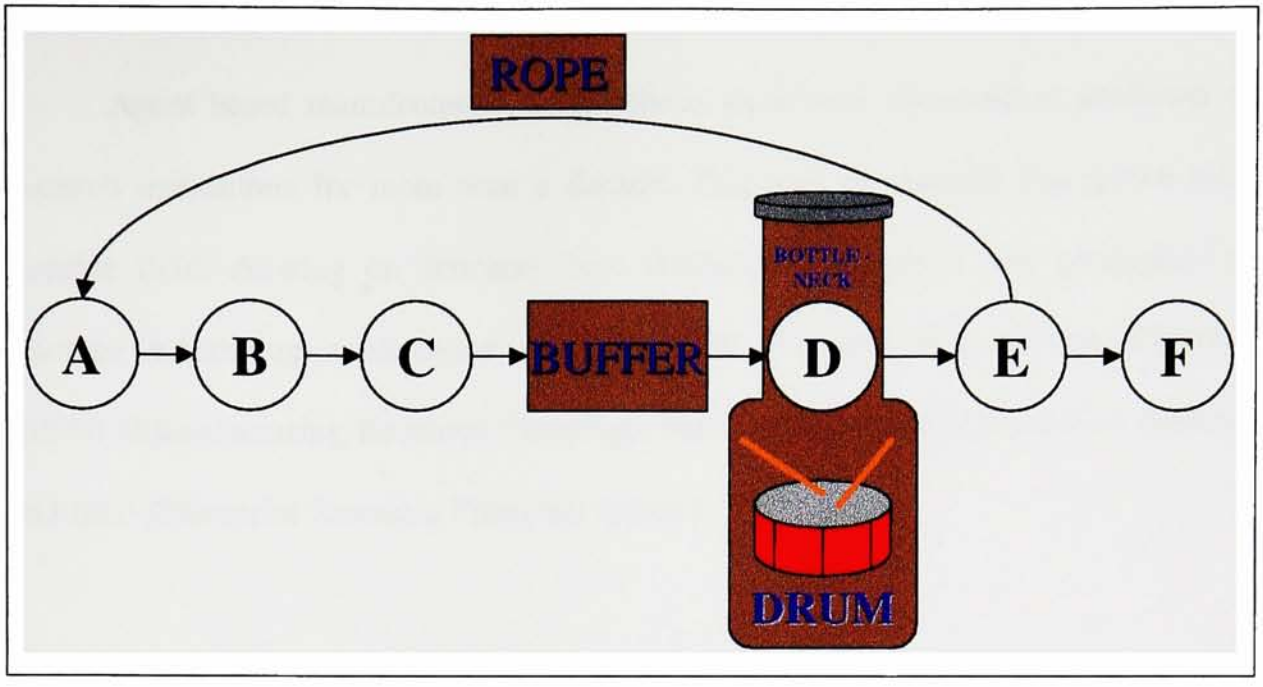


Figure 2-4 The Drum Buffer Rope Scheduling System

The bottleneck area (Operation D) drives the pace of the system; much like the drummer in a marching band drives the pace of the march. In order to assure that the bottleneck is never starved for work due to disruptions in upstream operations, a buffer is placed before the bottleneck. This buffer is referred to as a time buffer because it represents a set amount of processing time for the bottleneck, not a set number of parts. As units are processed through the bottleneck, operation E sends a signal back to operation A to start another part or batch. In this way the E to A communication is acting like a rope; every time Operation E tugs on the rope another part or batch is started at A. Operations E and F in this model would work like a pull system, matching product coming off the line to orders waiting in a shipping buffer, according to the customer delivery schedule (Kelly 18-19).

AGENT BASED MANUFACTURING ARCHITECTURES

Agent based manufacturing architectures have been discussed in academia and research institutions for more than a decade. This area of research has grown into a sizeable field, drawing on concepts from Artificial Intelligence, and production and resource scheduling applications such as, MRP (Material Requirements Planning), MRPII (Manufacturing Resource Planning), MES (Manufacturing Execution Systems), and ERP (Enterprise Resource Planning) systems.

AGENTS

An agent in a manufacturing architecture often is a software proxy that represents either a physical or functional element of a system (Shen and Norrie11). A functional agent represents functions or tasks in systems such as planning, scheduling, material handling, order acquisition, or product distribution. They tend to share many common state variables across different functions, which can lead to inconsistencies and unintended discourse between agents.

Physical agents commonly represent such physical objects as people, machines, vehicles, buildings, etc. Physical agents, by their nature, respond to a small set of variables that are within the control of individual agents and have minimal conversation between agents. (Shen and Norrie 11). Thus, functional agents tend to respond to and alter system knowledge, while physical agents tend to be more locally focused. This will be an important distinction in the following sections and chapters.

AGENT ARCHITECTURES

Due to the increasing complexity and communication of interconnecting shop floor control devices, architectures must be defined in a way that standardizes protocols for the interaction and integration of components. Two main architecture types have been proposed: Hierarchical and Heterarchical (Usher and Wangpara 2). Hybrid Architectures combine qualities of Hierarchical and Heterarchical Architectures and are discussed as a third option.

Hierarchical Architectures

Hierarchical Architectures can be thought of as a command and control system where decisions are made at the top levels and pushed downward toward the agents. Control within Hierarchical Architectures is divided into layers. Each layer is responsible for a different planning horizon and scope, resulting in a structure that resembles a pyramid. The extent or breadth of planning and scope decreases from top to bottom of the pyramid. Ottaway (11) discusses the Expanded AMRF Control Hierarchy (See **Figure 2-5**), which is a good example of Hierarchical Control. In this model command flows from the top Facility Agent downward as shown. Ottaway and Burns (490) state that the primary benefit of the Hierarchical Architecture is the ability to plan, and thus optimize, for conditions that may occur. Operational (short term) planning occurs at the lowest levels of the pyramid structure while tactical (long range) planning occurs at the higher levels.

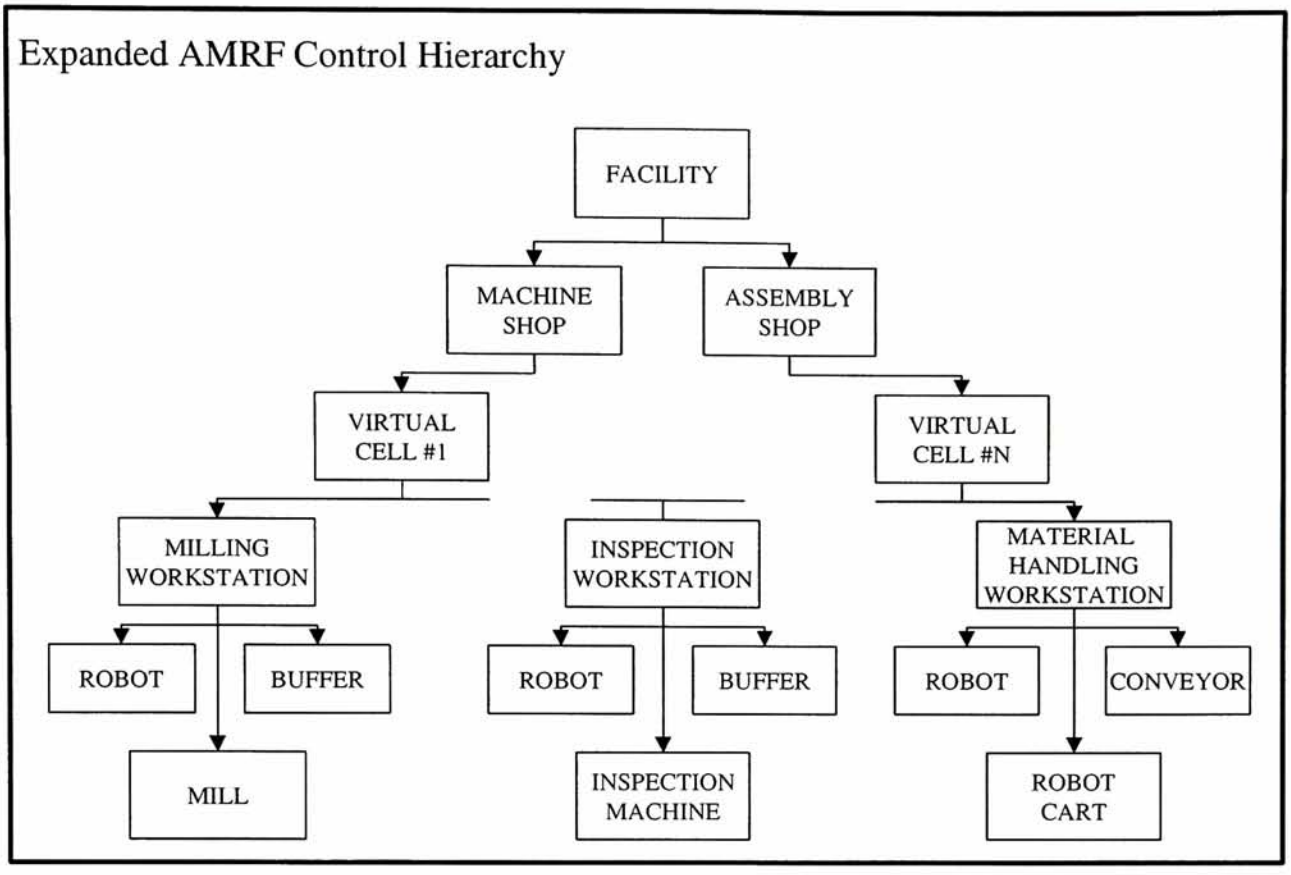


Figure 2-5 Hierarchical Control Architecture

Heterarchical Architectures

Heterarchical Architectures are known by several different names including Autonomous Agent, and Non-Hierarchical Architectures. As opposed to Hierarchical Architectures, Heterarchical Architectures tend to be flatter and distribute control between a few loosely coupled controllers (Usher and Wang para 3). A Heterarchical structure is one that provides flexibility and interaction between entities and/or agents, but avoids truly chaotic behavior by applying some well-formed simple behavioral rules (Lin and Solberg 57). Lin and Solberg also suggest that a Heterarchical model may more accurately represent the real world. Many different agent types act on their own behalf as they see fit to help achieve overall success of the system (61). Ottaway (11) presents the

following picture of what a distributed (Heterarchical) Control Architecture may look like in **Figure 2-6** (Adapted from Ottaway and Burns 490 and Ottaway 12).

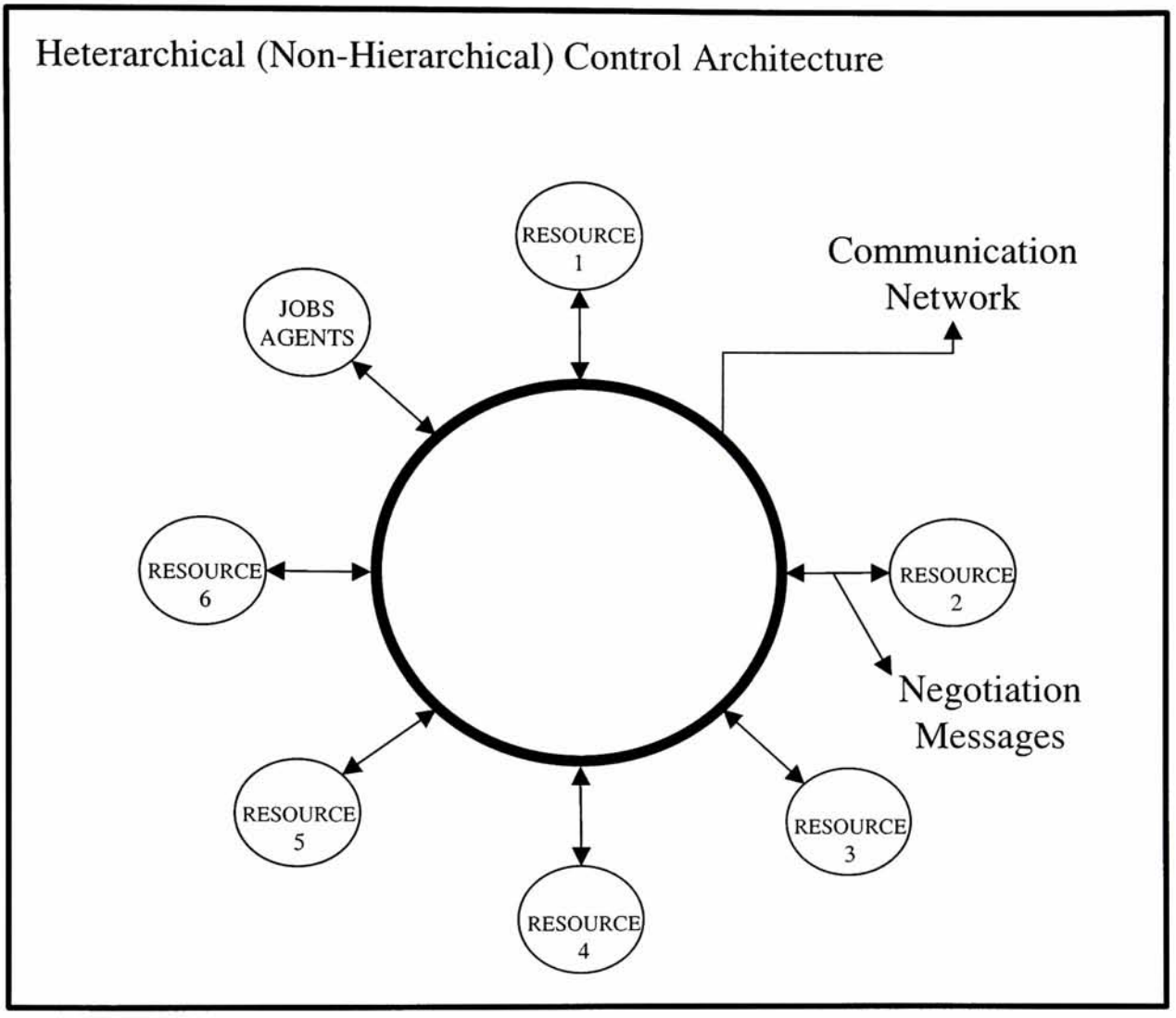


Figure 2-6 Heterarchical (Non-Hierarchical) Control Architecture

Shen and Norrie suggest four guidelines for what constitutes a Heterarchical (Autonomous Agent) Architecture (Shen and Norrie 13). This definition closely correlates to Axelrod and Choen's definition of an agent. According to Shen and Norrie, An Autonomous Agent:

1. is not controlled or managed by any other software agents or humans,

2. can communicate and interact directly with any other agents in the system and also with other external systems,
3. has knowledge about other agents and its environment, and
4. has its own goals and an associated set of motivations.

Hybrid (Federative) Architectures

Hybrid Architectures, also called Federative Architectures, are a compromise between Heterarchical and Hierarchical Structure. Federative Architectures tend to be innately Heterarchical, but can transition to Hierarchical when the local goals of its agents falter at a system level. Ottaway and Burns discussed the advantages of having an architecture that could flex between Hierarchical and Non-Hierarchical (or Heterarchical) coordination structure. They argued that a Non-Hierarchical structure has the benefits of reducing complexity of communication because the domains of control for these agents are more localized and there are fewer system variables to deal with (Ottaway and Burns 490). Hierarchical systems on the other hand have an advantage over Heterarchical systems in that they allow planning over an extended time period. The transition to Hierarchical structures within a system should be a function of the amount of planning required. (Ottaway and Burns 489-490).

Therefore, the ideal system architecture of Ottaway and Burns results in a structure that is balanced between a Heterarchical Architecture acting on local rules, and a Hierarchical Architecture emphasizing system level rules (489-490).

The ideal system:

- is innately Heterarchical,

- can be converted into a hierarchical system when the need for long term planning arises,
- creates levels of hierarchy as a function of the length of planning required,
- develops hierarchy within those agents that require planning, and
- is capable of returning to a Heterarchical state when the planning horizon is achieved.

Shen and Norrie (11) claim that Federative Multi-agent Architectures are superior to Hierarchical or Heterarchical Architectures alone. The improvement stems from the Federative Architecture's ability to coordinate multi-agent activity via facilitation as a means of reducing overheads, ensuring stability, and providing scalability.

Figure 2-7 illustrates the preceding manufacturing agent architecture discussion in light of Steven Wolfram's Complexity Classification.

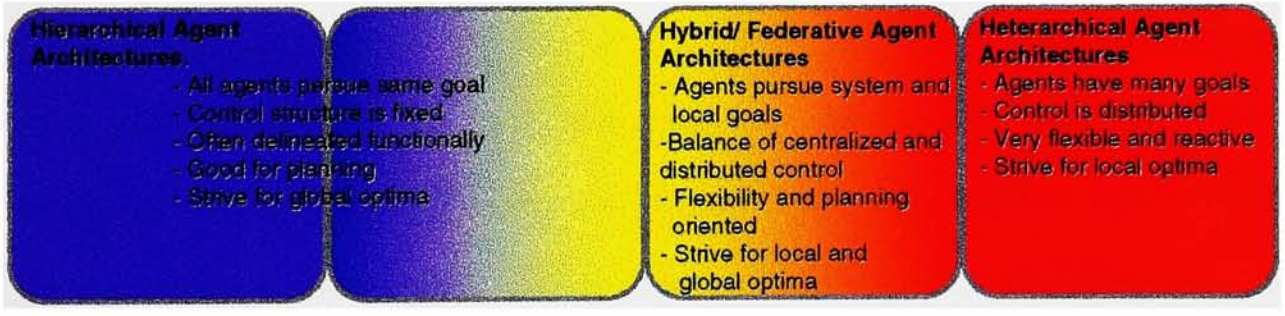


Figure 2-7 Agent Architectures in a Wolfram Classification

INTER AGENT BIDDING MECHANISMS

In their paper, Lin and Solberg use a market-like model coupled with an opportunistic negotiation scheme to balance system level and local level (agent) goals. The opportunistic negotiation focuses on principles of object-based architectures and allows intelligent entities (agents) to achieve individual goals (Lin and Solberg 59). The market-like model developed resembles an actual open marketplace. A job enters the system with some form of currency or worth and uses this currency to purchase services required to transform it into the final configuration. The job (Job Agent) bargains with resources (Resource Agents) to process it in a manner that meets the job's requirements. This can be accomplished using weighted objectives such as cost, time, and quality. Resources (Resource Agents), on the other hand, bid on jobs based on some internal measure to maximize their profit or some other system measure (Lin and Solberg 61).

Ottaway and Burns use a similar structure of system agents with system level goals and local level agents with parochial goals. There are three entity types in their model, two of them are local level Heterarchical Agents (Job and Resource Agents) and one is a system level Hierarchical Agent (Supervisor Agent).

Ottaway and Burns' Heterarchical Agents - Job Agents and Resource Agents

There are two types of Heterarchical agents in the Ottaway and Burns model, Job Agents, and Resource Agents. These agents communicate and negotiate via a multi-agent multi-bidding scheme similar to Lin and Solberg's. Job Agents in this model represent jobs coming into the system. The goal of the Job Agent is to procure the resources required to produce the part in the most (cost) efficient manner possible (Ottaway 65). The jobs come into the system at a predetermined rate, are assigned a Job Agent, and send out a request for bid to all the Resource Agents in the system. Job Agents seek to pay as little for the job, or processing, as possible. The goal of the Resource Agent is to obtain the processing of jobs to the resource it represents and to maximize each resource's utilization. (Ottaway and Burns 495-496). Both of these software proxies are physical agents, as they represent physical elements of the system that strive to attain parochial goals.

Ottaway and Burns' Hierarchical Agents - Supervisor Agents

The goal of the Supervisor Agent is to maximize the Throughput of the production system. The Supervising Agent's function is to balance between processing a job at low cost and high speed. A Resource Agent is assigned to a Supervising Agent if its utilization falls below a predetermined level. The Supervisor Agent helps improve the Resource Agent's utilization by assigning jobs to it. Once the utilization of the Resource Agent is at an acceptable level, the Supervisor Agent is deactivated.

Ottaway and Burns' Hybrid Agent Architecture

If after receiving the first set of bids, a Job Agent cannot find a Resource Agent to perform the job, it resubmits a request for new bids. If after three bid submissions the Job Agent still cannot find a suitable price and competency match, it increments its acceptable bid criteria, employing some pre-determined algorithms and parameters chosen by the modeler, and resubmits a new bid request. The Job Agent continues on this increasing bid cycle until a Resource Agent makes an acceptable bid, or until a Supervising Agent intervenes (Ottaway and Burns 495-496).

Once a Resource Agent is assigned to a Supervising Agent the process changes. If a Job Agent now requests a bid and there is a Supervising Agent representing a Resource Agent that can perform the job, the Supervisor Agent schedules the job directly. The Supervising Agent receives bids internally from the Resource Agent it supervises. The internal bids tell the Supervising Agent what jobs the Resource can perform, and when it can undertake them. If the Supervising Agent finds a match with one of its internal Resource Agents and a requested bid from a Job Agent, it schedules the bid with the Job Agent and the Resource Agent. If there are multiple Resource Agents under the Supervisor Agent that can perform the job, the bid is awarded to the Resource Agent that can start the job earliest (Ottaway 99-100). The Supervising Agent is a broker between the price of the job and the expedition of the job through the system. Once the Resource Agent's utilization increases, the Supervisor Agent relinquishes control. When the Supervisor Agent has no more Resource Agent's to control, it is disabled. The default state contains no active Supervisor Agents (Ottaway and Burns 499-500).

CHAPTER 3 MANUFACTURING THEORIES APPLIED TO RESOURCE ALLOCATION

There are three manufacturing theories simulated in conjunction with the research questions generated in Chapter 1. The first of the three concepts modeled is a static resource allocation system, the second concept modeled has both static and dynamic resource allocation capability, and the third concept has a completely dynamic resource allocation scheme.

THE STATIC BASE CASE METHODOLOGY

The static Base Case concept is very simple. In accordance with Wolfram Classes I & II discussed in Chapter 2, the goal of the Base Case manufacturing methodology is to create a very predictable and "Very Dull" or "Dull" production process (Shotwell). "Very Dull" and "Dull" are terms used by Shotwell to describe the lack of emergent properties in Class I and Class II systems. Chris Langton likened Class I systems to having Doomsday Rules which cause a system to settle into a predetermined outcome within a very short time period, much like a continuous flow manufacturing environment. Class II systems are only slightly more interesting; they may generate some basic patterns, but nothing emergent or out of the ordinary (Waldrop 225-226). An manufacturing example

might be a seasonal product with low variability in demand. Emergent properties are system behaviors that result from the interaction of components within the system, but would not occur if the components existed individually outside of the system. An example of an emergent behavior of interest in this research is Net Profit, a system performance measure described in Chapter 4.

The Base Case scenario has no dynamic resource allocation. This production system has one operator assigned to each Work Area (WA) and that assignment is static, meaning that the operator remains there for the duration of the simulated period. This production methodology is reminiscent of the stereotypical union shop where there are very specific job classifications and no job sharing.

THE SEMI-DYNAMIC THEORY OF CONSTRAINTS METHODOLOGY

The second manufacturing methodology modeled is intended to show elements of transition from Class I & Class II behaviors into Class IV behavior (see **Figure 2-2** in Chapter 1). Since the manufacturing methodologies chosen vary primarily on the means by which labor resources are allocated to perform work, a logical methodology to investigate is based on the Theory of Constraints (TOC) which was first introduced by Eli Goldratt in The Goal.

Goldratt developed The Goal as follows. In every system there is a goal (or a set of goals), toward which all functions in the system are assumed to be working. The TOC does not partition the system into individual components, but takes a holistic approach to improving the entire system (Kelly 14). In every system there are constraints or bottlenecks (the terms will be used interchangeably) that prevent the system from moving

closer to its goal(s). Thus, in a manufacturing setting, the constraint could, and very likely would be, a machine, machine operator, assembler, or process. These constraints can be physical, but need not be. For example, the constraint of a production system could be demand. If the demand for a product were low, according to Goldratt, then demand itself would become the constraint. To improve the manufacturing system, and move it closer to its goal(s), the systematic elimination of constraints becomes the driver of the TOC process (Kelly 14). This process is detailed in Chapter 2.

The goal for the TOC manufacturing model developed in this research is to achieve the highest Net Profit (See Chapter 2 for a definition of Net Profit). The TOC manufacturing model uses the same static resource allocation policy as the Base Case methodology. However, the TOC system has an additional pool of resources or assemblers that can be used to add capacity to bottleneck work areas. Additional capacity can be added to bottlneck Was at the end of each 8 hr period. The details of the TOC simulation model are found in Chapter 6.

THE DYNAMIC COMPLEX ADAPTIVE SYSTEM METHODOLOGY

The Complex Adaptive System (CAS) methodology is intended to elicit Class IV Wolfram behaviors (**Figure 1-1**). Richard Pascale's four qualifiers for CASs are used as a guideline for the development of the Complex Adaptive System (See Chapter 2 and **Figure 2-2**). A Hybrid, or Federative Agent Architecture, is developed using agents that carry out the different tasks of the Complex Adaptive System methodology. Communication between Heterarchical and Hierarchical Agents (working toward local and system goals respectively) within the system creates the balance required to allocate

the proper amount of resources to the proper locations at the proper time. It is important to note that the CAS manufacturing methodology differs from the TOC methodology in that there are many local level goals that each particular agent strives to fulfill, rather than an overriding system goal. These local goals act in concert to create the overall emergent system properties. The four agents employed in this manufacturing model include the Scheduler Agent, Assembler Agent, Job Agent, and Supervisor Agent.

Scheduler Agent

A Scheduler Agent is developed which is concerned only with arranging the incoming orders in order to reduce potentially large late penalties. The Scheduler Agent is a Heterarchical Agent because it strives for local optima and it represents a physical element of the system. It is a physical agent because it has complete control over the one task at hand and does the job a real life scheduler might do.

Assembler Agent

An Assembler Agent is developed to represent the workers in the manufacturing environment. These agents are concerned with assigning the resource they represents to a job and location that best meets the resource's preference. The Assembler Agents are considered Heterarchical Agents because they work toward obtaining local optima (Chapter 2 and **Figure 2-6**). They also are physical agents, because they represent assemblers in the system.

JOB AGENT

The Job Agents are a third Heterarchical Agent type developed representing physical elements of the system. They are a proxy for jobs as they enter the system, and continue to represent the job throughout all of the processing steps. The Job Agents are motivated to find an Assembler Agents to perform the next task in the job sequence at a low cost. They strive to fulfill objectives that are local (Chapter 2 and **Figure 2-6**) and represent physical elements of the system, namely actual jobs in the system.

SUPERVISOR AGENT

The last agent developed for the CAS methodology is the Supervisor Agent. It is a Hierarchical Agent representing functional elements of the system. It is a functional agent because it takes on the task of mediating between many different subsystems and agents. The specific functions of the Supervisor agent are detailed in Chapter 7. The Supervisor Agent has three tasks; it allocates, assigns, and either adds or removes Assembler Agents. The Supervisor Agent tracks many system level variables and springs into action only when needed, hence its Hierarchical nature (Chapter 2 **Figure 2-6**).

CHAPTER 4 MODEL SIMILARITIES

There are some common elements throughout the three manufacturing theories modeled in this research. The layout of the production process, the products produced, the order in which products arrive into the system, and some of the assumptions underlying each model are either identical, or nearly identical. The three methodologies differ primarily in the way assemblers are assigned to jobs. The details of what makes each model unique can be found in Chapters 5, 6, and 7.

The three different manufacturing theories are applied to an assembly line production system. Several other researchers have developed cellular or recursive manufacturing models. Lin and Solberg modeled an automated shop floor control system for a robotic manufacturing environment with an agent-based architecture (57-70). Ottaway and Burns examined a job shop environment where six different resources exist with various capabilities to undertake certain jobs (489-491). In both cases the jobs that entered the system could loop through several cycles of processing similar to jobs that flow through a job shop.

There exists little complexity or agent research in the area of manual assembly lines. It would be of interest, therefore, to apply these concepts to a manufacturing line.

The assembly line chosen is a variant of one developed by Anne Ostlund for her Vman virtual manufacturing game (Ostlund). Vman was developed as an educational tool for the introduction and application of Industrial Engineering (IE) concepts in an undergraduate IE program. The assembly line in her model is based on real case data pertaining to several variant electric hand held sanders.

PRODUCT SIMILARITIES

Each model produces the same four products, which can be thought of as variants of one another with module or subassembly differences across the different product types. Work Areas WA1, WA2, WA3, WA5, and WA7 can conceptually be considered subassembly Work Areas. The subassemblies are mated to the main assembly at WA4 and WA6. **Table 4-1** shows the variants of the products and the Work Areas required to assemble each variant. The values in the table represent the processing time in minutes for each product type at each WA. A gray cell denotes no subassembly or module is processed at that WA for that product.

PROCESS TIME ARRAY	WA1	WA2	WA3	WA4	WA5	WA6	WA7	WA8	WA9	WA10
Product 1	24	23	20	30		27	25	30	20	35
Product 2	12		12	18	14	17		18	12	21
Product 3	21	22		30		20	25	30	21	35
Product 4	14	15	14	20	16	19	17	21	14	24

Table 4-1 Process Time Array for All Models

LAYOUT SIMILARITIES

The basic layout for each of the manufacturing simulations is the same (See **Figure 4-1, 4-2, and 4-3**). Incoming orders wait in the Incoming Job Queue for processing. The capacity of the Incoming Job Queue is infinite, therefore allowing all jobs entering the system to queue without balking. There are ten Work Areas (WAs) where the processing steps of the products take place. Each WA has a capacity of two, meaning that up to two parts can be in a WA at any time, however, only one part can be processed at a WA at time unless additional personnel is added.

There are queue locations Q1, Q2, Q3, Q4, Q5 and Q7 after WA1, WA2, WA3, WA5, and WA7 respectively, each with a capacity of six. This queue capacity allows products with short processing time to proceed past products with long processing times. The ability to pass is crucial in the TOC and CAS models where adding assembler capacity allows more products to be produced, thus increasing the number of "fast" products that can potentially pass "slower" ones.

Finally, there are two staging areas Stage1 and Stage2. These are virtual locations where the simulation collects appropriate mating subassemblies or modules for the next processing step. Each staging area has a capacity of one.

From a product assembly perspective, there are no differences in the layout among the models. The models have a Home Location where additional non-utilized assemblers can wait for use. The CAS model has an additional location between the Incoming Job Queue and the Work Areas in order to force the simulation software to

accurately calculate required model data. This extra location has a capacity of one and is essentially an extension of the Incoming Job Queue

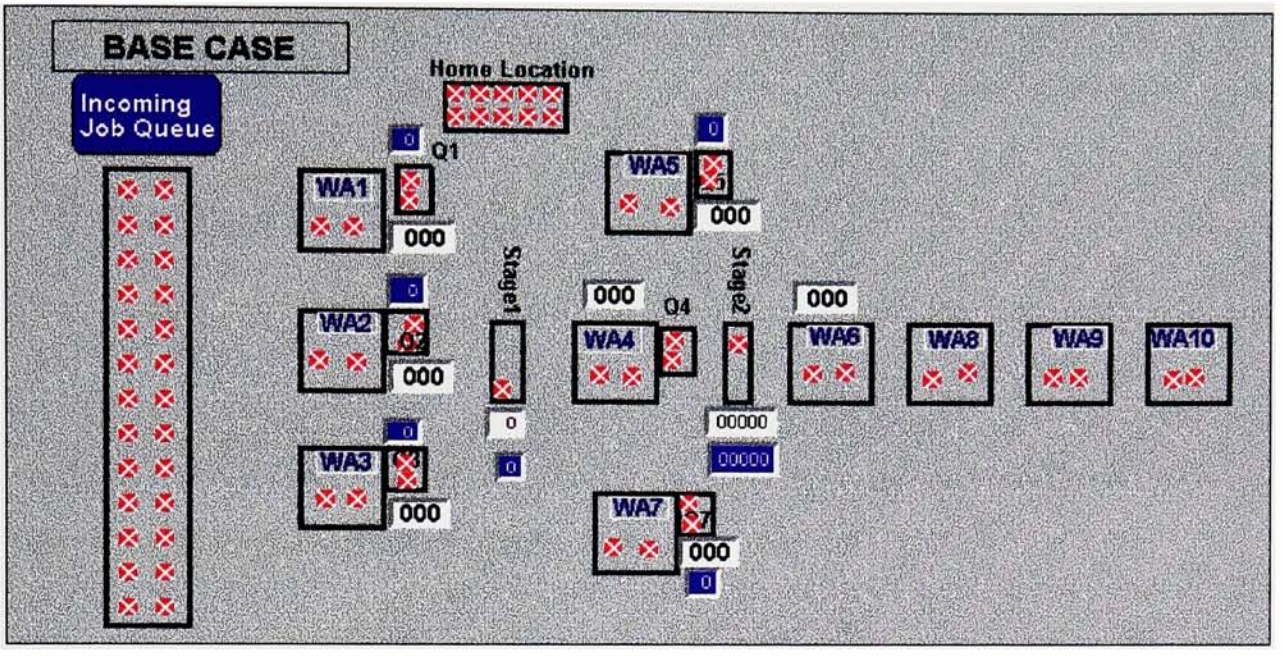


Figure 4-1 Base Case Layout

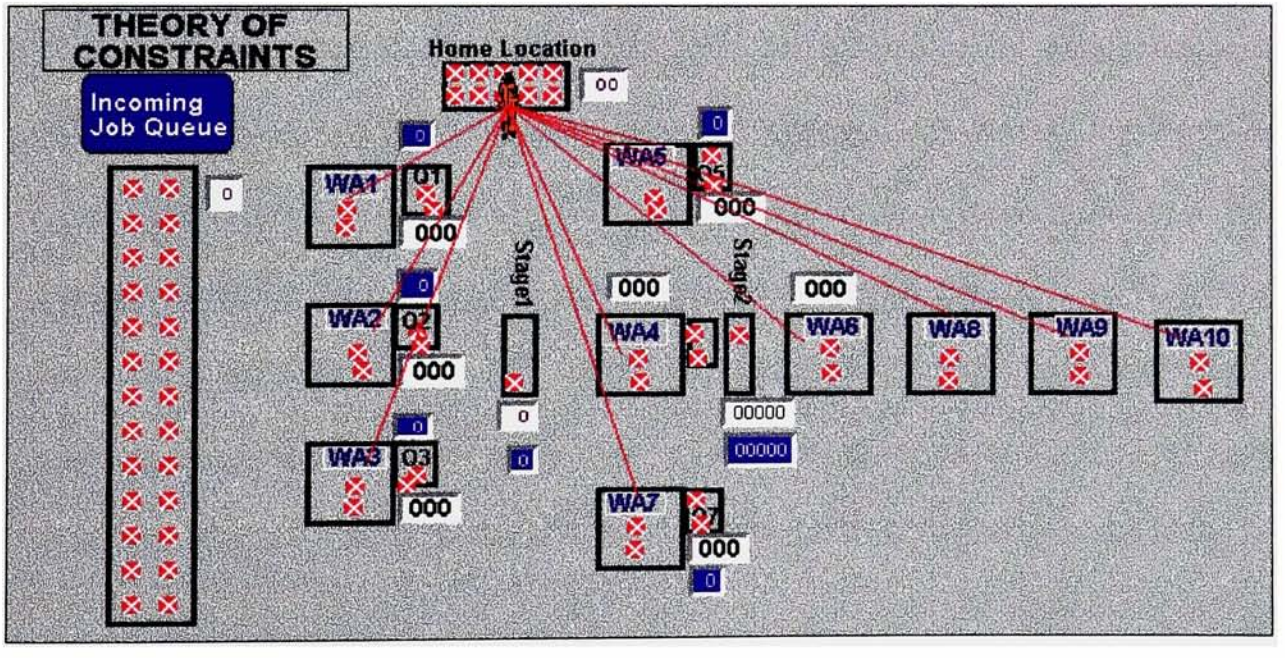


Figure 4-2 Theory of Constraints Layout

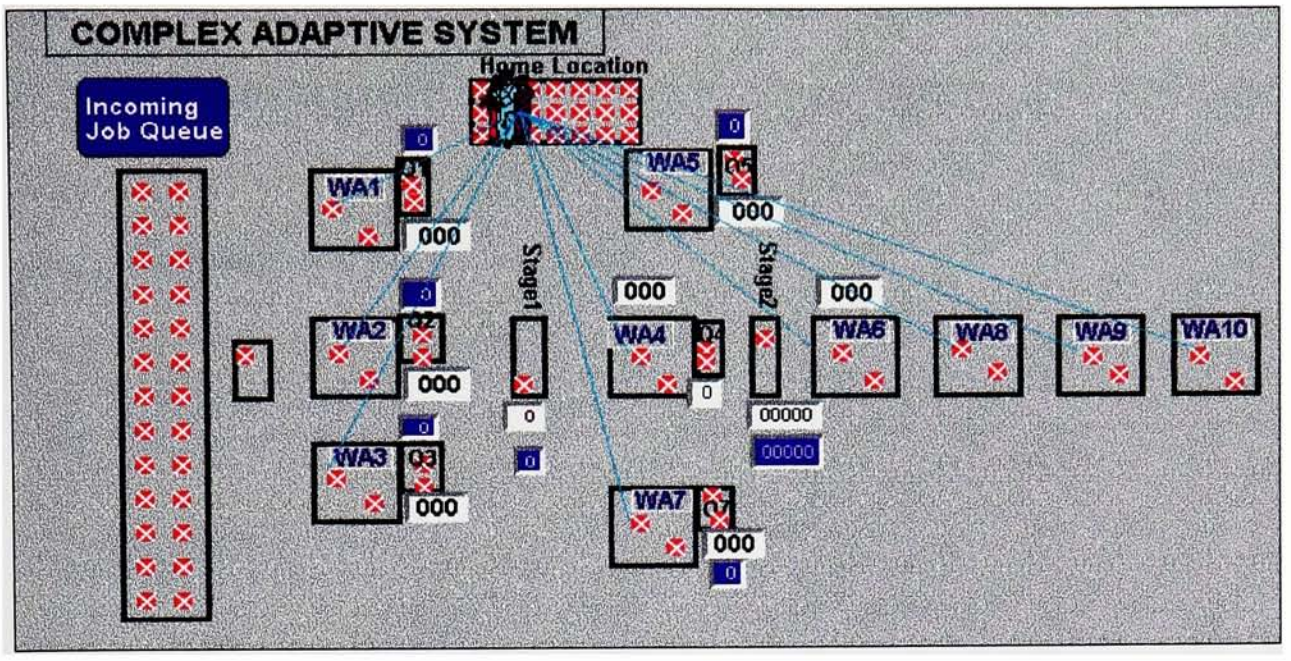


Figure 4-3 Complex Adaptive System Layout

ORDER ARRIVAL SIMILARITIES

Each model uses the same set of three arrival files representing three different paces: fast, medium, and slow (these will also are referred to as highly stressed, moderately stressed, and lightly stressed systems). Orders arrive at time increments of twenty minutes on average for the fast pace (highly stressed system), twenty-seven minutes on average for the medium pace (moderately stressed system), and thirty-four minutes on average for the slow case (lightly stressed system). The product sequence is randomized, as is the size of the order. The order of arrival is exactly the same for each arrival pace, as is the size of the order, which varies uniformly, between one and five.

Tables 4-2, **4-3**, and **4-4** are portions of fast, medium, and slow paced arrival files. Notice that in each file the incoming order of products is identical. The timing of arrivals is equally spaced out (i.e. they arrive every sixty, eighty-one, or one hundred and two minutes), and the average number of orders arriving at one time is three. Therefore,

on average, the arrivals enter into the system at three times the time between arrivals (20, 27, and 34 minutes per arrival respectively). Multiple quantities can arrive at once, but only one product type can enter per arrival. An attribute of each order, ID_AT, is a unique identifier used to join sub assemblies.

entity name	location name	quantity per arrival	time of first arrival	number of arrivals	PROD_AT	random nuber	DUE_DATE_AT	ID_AT
Prod	IncJobQ	1	0	1	1	0.795267663	360	1
Prod	IncJobQ	1	0	1	1	0.853333056	360	2
Prod	IncJobQ	1	0	1	1	0.235173812	360	3
Prod	IncJobQ	1	0	1	1	0.448021978	360	4
Prod	IncJobQ	1	0	1	1	0.014788382	360	5
Prod	IncJobQ	1	60	1	3	0.106384631	420	6
Prod	IncJobQ	1	120	1	3	0.172069551	480	7
Prod	IncJobQ	1	120	1	3	0.041743279	480	8
Prod	IncJobQ	1	120	1	3	0.328869811	480	9
Prod	IncJobQ	1	180	1	1	0.278496848	540	10
Prod	IncJobQ	1	180	1	1	0.734555256	540	11
Prod	IncJobQ	1	240	1	1	0.704054906	600	12
Prod	IncJobQ	1	300	1	2	0.67961437	660	13
Prod	IncJobQ	1	300	1	2	0.266924782	660	14
Prod	IncJobQ	1	300	1	2	0.574183443	660	15
Prod	IncJobQ	1	360	1	3	0.141774041	720	16
Prod	IncJobQ	1	360	1	3	0.658887124	720	17
Prod	IncJobQ	1	360	1	3	0.252667504	720	18
Prod	IncJobQ	1	360	1	3	0.424706378	720	19
Prod	IncJobQ	1	360	1	3	0.903595701	720	20

Table 4-2 Highly Stressed Arrival File

entity name	location name	quantity per arrival	time of first arrival	number of arrivals	PROD_AT	random nuber	DUE_DATE_AT	ID_AT
Prod	IncJobQ	1	0	1	1	0.929327907	360	1
Prod	IncJobQ	1	0	1	1	0.742054577	360	2
Prod	IncJobQ	1	0	1	1	0.618728309	360	3
Prod	IncJobQ	1	0	1	1	0.532777457	360	4
Prod	IncJobQ	1	0	1	1	0.241148707	360	5
Prod	IncJobQ	1	81	1	3	0.077184586	441	6
Prod	IncJobQ	1	162	1	3	0.475090709	522	7
Prod	IncJobQ	1	162	1	3	0.601493103	522	8
Prod	IncJobQ	1	162	1	3	0.660664919	522	9
Prod	IncJobQ	1	243	1	1	0.236477888	603	10
Prod	IncJobQ	1	243	1	1	0.516446915	603	11
Prod	IncJobQ	1	324	1	1	0.941075768	684	12
Prod	IncJobQ	1	405	1	2	0.689006186	765	13
Prod	IncJobQ	1	405	1	2	0.510078167	765	14
Prod	IncJobQ	1	405	1	2	0.220578225	765	15
Prod	IncJobQ	1	486	1	3	0.790755448	846	16
Prod	IncJobQ	1	486	1	3	0.464268346	846	17
Prod	IncJobQ	1	486	1	3	0.241141731	846	18
Prod	IncJobQ	1	486	1	3	0.058551045	846	19
Prod	IncJobQ	1	486	1	3	0.897245415	846	20

Table 4-3 Moderately Stressed Arrival File

entity name	location name	quantity per arrival	time of first arrival	number of arrivals	PROD_AT	random nuber	DUE_DATE_AT	ID_AT
Prod	IncJobQ	1	0	1	1	0.606971905	360	1
Prod	IncJobQ	1	0	1	1	0.28240716	360	2
Prod	IncJobQ	1	0	1	1	0.732044064	360	3
Prod	IncJobQ	1	0	1	1	0.616076942	360	4
Prod	IncJobQ	1	0	1	1	0.702977429	360	5
Prod	IncJobQ	1	102	1	3	0.152659771	462	6
Prod	IncJobQ	1	204	1	3	0.482933164	564	7
Prod	IncJobQ	1	204	1	3	0.097930646	564	8
Prod	IncJobQ	1	204	1	3	0.988204048	564	9
Prod	IncJobQ	1	306	1	1	0.363282403	666	10
Prod	IncJobQ	1	306	1	1	0.007807251	666	11
Prod	IncJobQ	1	408	1	1	0.88633524	768	12
Prod	IncJobQ	1	510	1	2	0.909720504	870	13
Prod	IncJobQ	1	510	1	2	0.576397787	870	14
Prod	IncJobQ	1	510	1	2	0.013989361	870	15
Prod	IncJobQ	1	612	1	3	0.600838765	972	16
Prod	IncJobQ	1	612	1	3	0.048833341	972	17
Prod	IncJobQ	1	612	1	3	0.803567608	972	18
Prod	IncJobQ	1	612	1	3	0.04880094	972	19
Prod	IncJobQ	1	612	1	3	0.485360949	972	20

Table 4-4 Lightly Stressed Systems

SYSTEM MEASURES

The System Measures for all three models include: Amount of Potential Sales (Sales Dollars) Earned, Total Late Penalties, Total Throughput, Total Resource Cost, and Net Profit. These measures are derived from the Theory of Constraints (Kelly 16-18 and Chapter 2) and are detailed in **Equation 4-1**, **4-2**, and **4-3**.

Equation 4-1: Net Profit = Throughput - Operational Expenses

Equation 4-2: Throughput = (Sales Revenue) - (Materials Costs) - (Late Penalties)

Equation 4-3: Operational Expenses = (Overhead) + (Labor Costs) + (Scheduling Costs)

Each of these Measures will be described below:

SALES DOLLARS

Sales Dollars is a measure that represents the highest potential Throughput a product can have, assuming no late penalties. The sales prices for all products are as follows: $600 for Product 1, $350 for Product 2, $250 for Product 3, and $500 for Product 4.

LATE PENALTY

Late penalties are assessed as a prorated percentage of the Sales Dollars for every minute the product is late. The late penalty is calculated at ten percent of the Sales Dollar value per day, prorated to the minute. Once a product is at least one minute late it begins accruing a late penalty proportional to ten percent of the Sales Dollars per day. The Total Late fee is accumulated over the system run.

TOTAL THROUGHPUT

Total Throughput is calculated as the sum of Throughput from **Equation 4-2** (Kelly 16).

TOTAL COST OF RESOURCES

The Total Cost of Labor is calculated by multiplying the hourly cost of each operator by the total hours of the system run. If an operator is ever used, then its presence is assumed to be required for the entire run of the system. Otherwise, it would be impossible to predict how many resources would be required at peak demand periods.

NET PROFIT

Net Profit is an all-encompassing system measure derived from the TOC equation, where Net Profit equals Throughput minus Operating Expense (See **Equations 4-1, 4-2**, and **4-3**). It is also a very tangible real world measure used frequently in industry.

COMPLEXITY (POWER LAW) MEASURES

Six Power Law measures are tracked for each type of model. The results of these measures give an indication of whether or not a system is displaying complex behavior. The six measures are: Late Fee Per Late Job, Percent of Sales Per Late Job, Time Between Late Exits, Late Jobs Per Day, Throughput Per Day, and Maximum Queue Length Per Day. Histograms of the data output for each measure are built, and cumulative distributions of the data are plotted on log-log scale. A Power Law equation is fit to each data set. Data sets that have a "good fit" to the Power Law equations represent systems with Self-Organized Criticality or Complexity. These measures are used to answer research question two.

ASSUMPTIONS

Several assumptions are applicable to all of the models and are listed below.

It is assumed that:

- Overhead is a fixed cost and is the same for each model, thus its effect is negligible and is ignored.
- Materials Costs are fixed and they have been directly deducted from the Potential Sales Cost per product.
- Customers accept partial shipments. This essentially means that only the late products per multiple job order are penalized, not the entire shipment.
- Model run length of 160 hours represents one month of processing for an 8-hour shift at five days a week.
- The system represents a build to order manufacturing environment, where every arrival that comes into the system has a demand associated with it.
- No setup time or cost is incurred for new product types due to similarities in configurations.

CHAPTER 5 UNIQUE ELEMENTS OF THE BASE CASE MODEL

MODEL TO MODEL VARIATION

The three scenarios simulated for this thesis vary in their means of assigning assemblers to jobs. Resource allocation in the Base Case model is strictly static and is detailed later in this chapter (See **Table 5-1**). The Base Case serves as a starting point for the TOC and CAS models. This platform is discussed in the Justification for the Base Case model section below. Arrow diagrams are used to differentiate between manufacturing models, and to aid in the identification of System Measures and their interactions.

RESOURCE ALLOCATION

The resource allocation of the Base Case model is very simplistic. One resource, or operator, or assembler (the terms are interchangeable) is assigned to work solely at one of the ten Work Areas (WAS) throughout the entire simulated period.

	Static Resource Allocation	Dynamic Resource Allocation
Base Case	X	
Theory of Constraints		
Complex Adaptive System		

Table 5-1 The Resource Allocation of the Base Case Model Is Strictly Static

ARROW DIAGRAMS - JUSTIFICATION FOR THE BASE CASE MODEL

Arrow diagrams such as **Figure 5-1** are useful tools, which allow system modelers to anticipate performance the system they are designing by identifying key elements of the system and the interactions among key elements. A positive (+) sign at the head of an arrow indicates that the element at the tail of the arrow has a reinforcing affect on the element at the head. A negative (-) sign at the head of an arrow indicates that the element at the tail of the arrow has an undermining or weakening affect on the element at the head. Closed arrow loops that have a negative net sign (multiplying all the signs of the arrows results in a negative) are control loops. A positive net sign on a loop are destabilizing. Identifying control loops and the interaction and competition between all loops aids system prediction (Stiebitz 16).

Although there are no loops in the Base Case system, the arrow diagram is still useful for identifying system interactions. For example, if Operating Expense is increased, it has an undermining or weakening effect on Net Profit. Conversely, if Operating Expense is decreased, it increases Net Profit. Also, as Throughput is

increased, Net Profit is increased, and as Throughput decreases, Net Profit Decreases. **Table 5-2** details the exact interactions of all the elements.

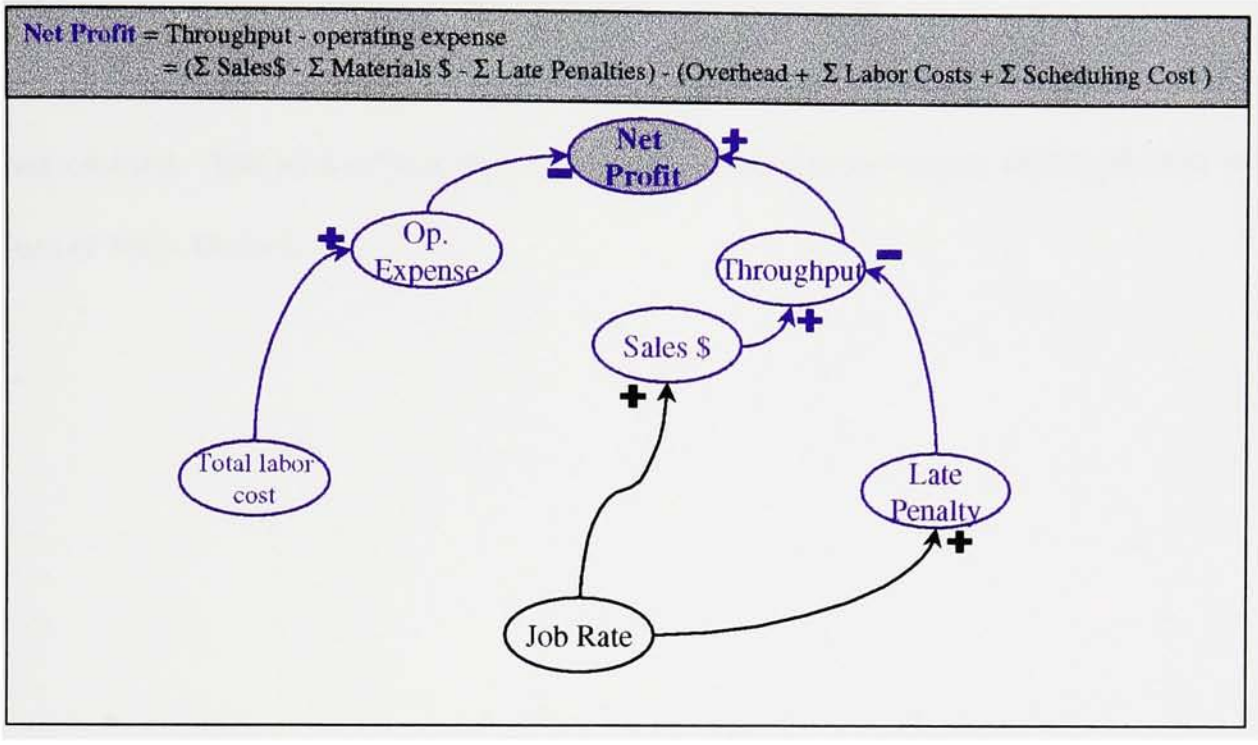


Figure 5-1 Base Case Arrow Diagram

BASE CASE		
From	**To**	**Description**
Job Rate	**Sum of Sales Dollars**	As more jobs enter the system there will be a greater potential for increased sales revenue
Job Rate	**Sum of Late Penalties**	The more jobs in the system, the greater chance of incurring a late penalty due to capacity constraints
Sum of Late Penalties	**Throughput**	As the overall Late Penalties increase, the system Throughput($) decreases
Sales Dollars	**Throughput**	The more Sales Dollars generated, the more Throughput generated based on the Net Profit equation
Throughput	**Net Profit**	Increasing Throughput increases Net Profit (See Net Profit equation)
Sum of Labor Costs	**Sum of Operational Expenses**	Higher Sum of Labor Costs drives higher Operational Expenses based on the Net Profit equation
Sum Operational Expense	**Net Profit**	Increasing operating expenses decreases net profit (See Net Profit equation)

Table 5-2 Base Case Arrow Diagram Definitions

In **Figure 5-1**, notice that all of the elements of the Net Profit equation are not active elements of the Base Case system. There is no scheduling conducted in the Base Case, and thus no scheduling cost is incurred. The sum of scheduling cost will only be a factor in the Complex Adaptive System model (see Chapter 7).

Overhead, as stated in Chapter 4, is assumed to be the same for each model and is thus omitted. The sum of materials cost is also assumed previously deducted from the Sum of Sales Dollars.

CHAPTER 6 THE THEORY OF CONSTRAINTS MODEL

RESOURCE ALLOCATION

The resource allocation of the Theory of Constraints model is more dynamic than the Base Case. The allocation of assemblers is derived from the Theory of Constraints Process (Kelly 18, and Chapter 2). There are static resources assigned to each Work Area (WA) as in the Base Case model (**Table 6-1**). The first ten assemblers are initially allocated to their Work Areas and remain there for the entire duration of the simulation. Unlike the Base Case, however, the TOC model allows additional capacity, in the form of an additional resource, to be added to any WA (**Table 6-1**). These additional resources can also be removed but the original static resource at each WA cannot be removed.

	Static Resource Allocation	Dynamic Resource Allocation
Base Case	X	
Theory of Constraints	X	X
Complex Adaptive System		

Table 6-1 TOC Semi-Dynamic Resource Allocation

RUNNING THE TOC MODEL

The Theory of Constraints model is run for eight hours at a time and then manipulated by the Intervention Agent, which is the system modeler. Upon the completion of eight hours of run time, the Intervention Agent evaluates the utilization of each Work Area as an indicator of bottlenecks and then determines whether to add or remove an assembler. The rules that determine resource allocation are listed below.

WHEN TO ADD CAPACITY (EXTRA ASSEMBLERS)

Additional capacity is required in the TOC model at areas that become worthy constraints or bottlenecks. At the end of every eight-hour interval the Intervention Agent considers the WA with the highest utilization to be the constraint. This bottleneck WA is a worthy bottleneck if its WA utilization is greater than 90%. If the worthy bottleneck WA identified has only a static assembler assigned to it, then an additional resource will be assigned to double the capacity. The assignment of additional assemblers is only done at the end of an eight-hour shift. Only the worthy bottleneck can receive an

additional resource per eight-hour shift, and keeps its extra capacity until such time it is to no longer needed. Only one WA can receive an extra resource per day.

When to remove capacity (Extra Assemblers)

The Intervention Agent's removal of resources, likewise, occurs only at the end of an eight-hour shift. Any WA with extra capacity at the end of an eight-hour period, and with a WA utilization less than 50%, has the extra assembler removed. Unlike the adding of capacity, multiple resources can be removed from multiple Work Areas at one time.

Special Case of the Resource Scheduler: Drum Buffer Rope

There is one unique instance when a Drum Buffer Rope (DBR - See Chapter 2) sequencing of products is required. Stations downstream from WA4 are not affected by altering the release of material from the Incoming Job Queue due to the rapid nature of bottleneck change and amount of processing time between the downstream WAs and the beginning of processing. If Work Area 4 (WA4) is determined to be the worthy bottleneck for a given eight-hour period, then a DBR release of products out of the Incoming Job Queue is required. This DBR sequencing enables subordination, step three of the Theory of Constraints process (Goldratt 301- 307; Kelly 18). Subordinating Work Areas one through three by limiting the release of products from the Incoming Job Queue based on the production rate of WA4 (the bottleneck) prevents work from piling up in the system. If WA4 is identified as the bottleneck, it initiates the DBR sequence for one day prior to the addition of an additional resource, so that subordination of upstream job release can be accomplished. If after one day WA4 is still the worthy bottleneck, it is

assigned an extra assembler, and DBR sequencing continues while WA4 remains the bottleneck. As soon as another WA becomes the bottleneck, DBR sequencing is terminated. WA4 behaves exactly the same as any other station for removal of capacity.

JUSTIFICATION FOR THE THEORY OF CONSTRAINTS MODEL

The arrow diagram in **Figure 6-1** is the core for the justification for the Theory of Constraints model. As in the Base Case model, there are no instances of looping, but there is still value in the mapping. Notice the additional elements of this model, primarily the Intervention Agent elements. This Intervention Agent is symbolic of the modeler who acts as a personnel supervisor, adding and removing capacity (assemblers). The From-To Relationships, **Table 6-2**, follows the Arrow Diagram and details the relationship between each element of **Figure 6-1**.

SYSTEM MEASURES

The arrow diagram (**Figure 6-1**) gives insights to some of the interactions between the model and measures of the system. Some elements of the Net Profit equation are still not applicable. There is, however, an additional resource cost representing the Intervention Agent's Salary. The modeler acts by adding and removing resources, much like a personnel supervisor on a manufacturing line. Therefore, a salary equivalent to that of a supervisor's is incurred for the Intervention Agent. There are no queue scheduling costs associated with the TOC model, because there is no scheduling agent, and thus no rearrangement of the queue.

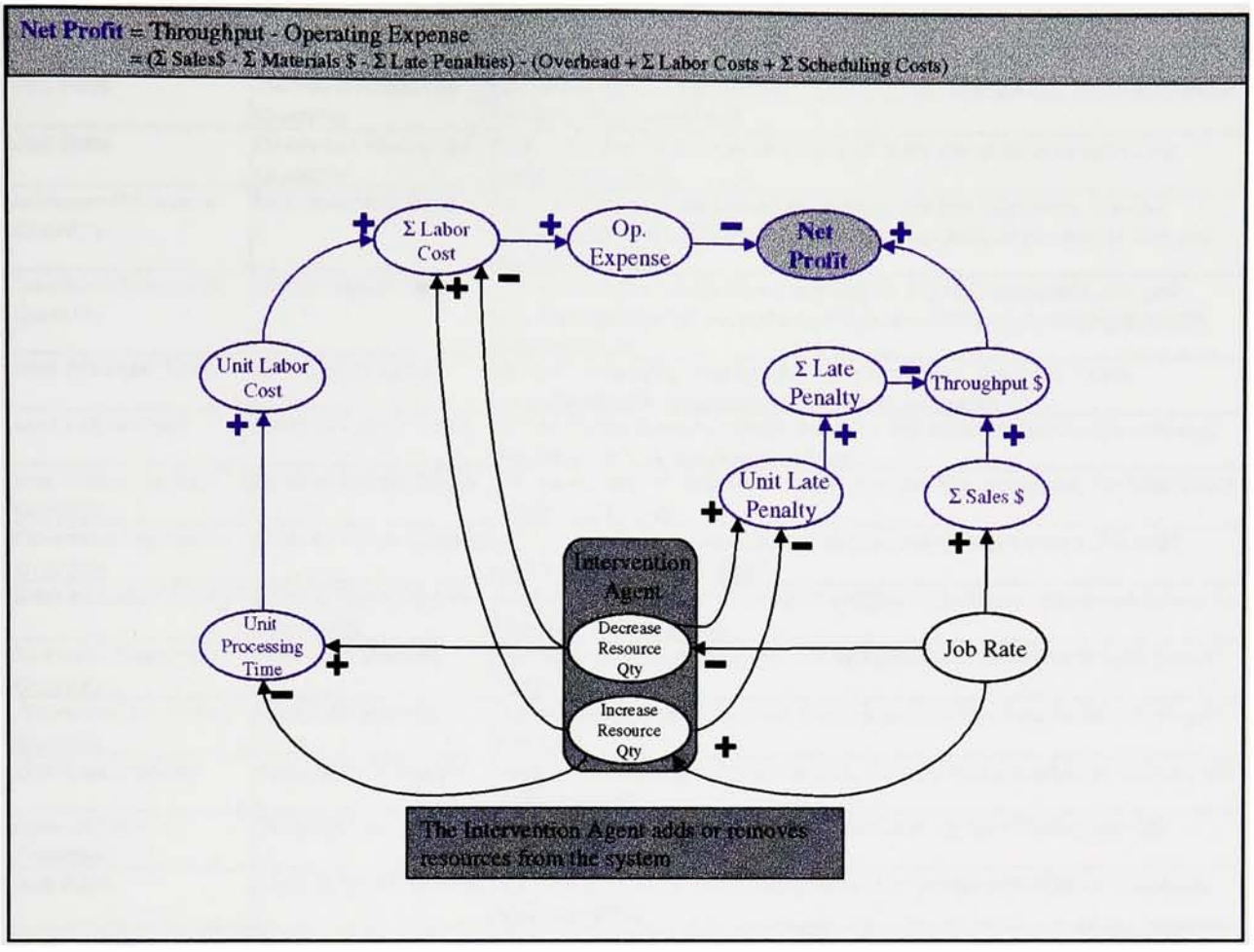


Figure 6-1 The Theory of Constraints Arrow Diagram

Theory of Constraints		
From	**To**	**Description**
Job Rate	**Increase Resource Quantity**	The more jobs in the system, the greater the chance that more resources will be allocated to work
Job Rate	**Decrease Resource Quantity**	The more jobs in the system, the less likely the chance of removing resource capacity
Increase Resource Quantity	**Unit Process Time**	As the number of resources working on the line increases, the unit processing time will decrease and thus the Costs of producing that unit also decrease
Decrease Resource Quantity	**Unit Process Time**	As the number of resources working on the line decreases, the unit processing time will increase and thus the Costs of producing that unit also increase
Unit Process Time	**Unit Labor Cost**	As the processing time for each unit increases, the labor Costs associated with producing that unit also increases
Unit Labor Cost	**Sum of Labor Costs**	As the Costs associated with the labor allocated to each unit increases, the total labor Costs also increases
Increase Resource Quantity	**Sum of Labor Costs**	As the number of resources working on the line increases, the total Costs of labor will go up
Decrease Resource Quantity	**Sum of Labor Costss**	As the number of resources working on the line decreases, the total Costs of labor will go down
Sum of Labor Costs	**Sum of Operational Expenses**	Higher Sum of Labor Costs drives higher Operational Expenses based on the Net Profit equation
Increase Resource Quantity	**Unit Late Penalty**	As more resources are used, the late penalty incurred for the job should decrease
Decrease Resource Quantity	**Unit Late Penalty**	As fewer resources are used, the late penalty incurred for the job should increase
Unit Late Penalty	**Sum of Late Penalty**	As each unit late penalty increases, the total late penalties for all jobs will also increase
Sum of Late Penalties	**Throughput**	As the overall Late Penalties increase, the system Throughput ($) decreases
Job Rate	**Sum of Sales Dollars**	As more jobs enter the system there is a greater potential for increased sales revenue
Sum of Sales Dollars	**Throughput**	The more Sales Dollars generated, the more Throughput generated based on the Net Profit equation
Sum Operational Expense	**Net Profit**	Increasing operating expenses decreases Net Profit (See Net Profit equation)
Throughput	**Net Profit**	Increasing throughput increases Net Profit (See Net Profit equation)

Table 6-2 From To Definitions for the TOC Arrow Diagram

ADDITIONAL ASSUMPTIONS OF THE THEORY OF CONSTRAINTS

Beyond the generic assumptions for all the models, the TOC model also assumes:

- WA utilization equals the time spent on processing product divided by the current system run-time.
- WA utilization is a valid indicator of system bottlenecks.
- A Work Area never needs more than one extra resource.
- Only one constraint can be identified per day.
- Only one resource can be added per day.
- Multiple resources can be removed in one day, returning the respective WAs to one resource each.
- WA1, WA2, and WA3 work in parallel and essentially are always pulling the next product from the Incoming Job Queue once they become available.
- Adding a resource doubles the WA capacity and allows the area to produce on average twice as much.
- An additional cost of the Intervention Agent billed at $30.00 per hour over the entire run of the model.

CHAPTER 7 THE COMPLEX ADAPTIVE SYSTEM MODEL

The Complex Adaptive System (CAS) model uses a Federative Agent architecture to assign assemblers to Work Areas. There are four agent types used in the CAS model, the Scheduler Agent, Assembler Agents, Job Agents, and a Supervisor Agent. The architecture is Federative because it, by default, uses three local level physical agents (Scheduler, Assembler, and Job Agents) and when needed, one system level functional agent (Supervisor Agent). Refer to Chapter 2 for a definition of local level physical agents, and system level functional agents.

The Scheduler Agent's goal is to minimize the potential late penalty by promoting high-risk jobs in the Incoming Job Queue. The Assembler Agents represent the assemblers in the system, and the Job Agents represent the jobs that come into the system. The Assembler and Job Agents go through a bidding routine to decide which assemblers work at a given Work Area. This bidding routine is detailed later in this chapter. Finally, when needed, the Supervisor Agent performs three system level functions, allocation of assemblers, assigning assemblers, and adding or removing assemblers.

SCHEDULER AGENT

The Scheduler Agent arranges the jobs in the Incoming Job Queue in an effort to avoid potentially high late penalties. The Scheduler Agent is physical agent that represents a real life scheduler. It acts very parochially, with no direct interaction with any other agents, although the Scheduler Agent does have indirect interaction with the Supervisor Agent. Unlike the Assembler and Job Agents, which have multiple manifestations, there is only a single instance of the Scheduler Agent.

SCHEDULER AGENT'S GOAL AND MEANS OF ACHIEVING IT

The Goal of the Scheduler Agent is to arrange the jobs in the Incoming Job Queue in an effort to avoid high cost late fees based on the following set of rules. The Scheduling Agent promotes jobs in the queue based on their Sales Dollars. The high dollar products are therefore promoted slightly over their lower cost counterparts because of their potential for a larger late fee (See **Table 7-1**).

	Sales Price
Product 1	$ 650.00
Product 2	$ 350.00
Product 3	$ 250.00
Product 4	$ 500.00

Table 7-1 Sales Dollars (Sales Price) per Product

Product 1, having the highest sales value of $650, can preempt all other products in the queue, but cannot preempt another product 1. It can preempt up to three other products, but once it passes three it cannot move any further in the queue. This prevents product 1's from completely dominating the queue. Product 4 (sales value of $500) can preempt a product 2 or a product 3 up to two times. It cannot preempt another product 4, or a product 1. Product 2 (sales value of $350) can only preempt a product 3, and it can only preempt once before it must stop. Product 3 (sales value of $250) is the lowest Sales Dollar product and cannot preempt any other products. These basic rules govern the activity of the Scheduler Agent. Each change in the queue order incurs a cost of $10.00. The sum of queue arrangement costs is tallied for two hours at a time and this amount is used by Supervisor Agent to make allocation decisions.

ASSEMBLER AGENT

The Assembler Agents represent the real assemblers in the system. There is an Assembler Agent that represents each human resource working on the assembly line. They are local and physical agents because they represent a physical resource and have limited interaction with other agents in the system (Shen and Norrie 11). There are three types of Assembler Agents in the CAS model, Alpha, Beta, and Gamma, which symbolize different labor classifications. Alpha Agents can work at any Work Area. Beta Agents can only work at WA1- WA5, and Gamma Agents can only work at WA6-WA10. The pay rate for each type of agent is reflective of the type of work they can do. Alpha Agents are the most versatile, so the pay rate associated with Alpha Agents is the highest. Gamma Agents have the second highest pay rates because they work the second

half of the assembly line and would have Final Run and Test responsibilities as a part of their job function. Beta Agents work at WAs at the first half of the assembly line, and have the lowest pay rate of the three labor classifications. Appendix B details Assembler Agent pay rates and the eligible Work areas for each Assembler Agent.

ASSEMBLER AGENTS' GOAL AND MEANS OF ACHIEVING IT

Each Assembler Agent has a local goal, namely, to maximize its utility. Utility is a measure of how much an Assembler Agent prefers a Work Area. It is also a measure of the likelihood that an Assembler Agent bids for a job at a given Work Area. Each Assembler associates a utility curve to each eligible Work Area. Assembler Agents use these utility assignments in their negotiation with Job Agents. **Figure 7-1** shows three different utility curves. Assembler Agents choose a curve that best matches their interest to work at a particular WA.

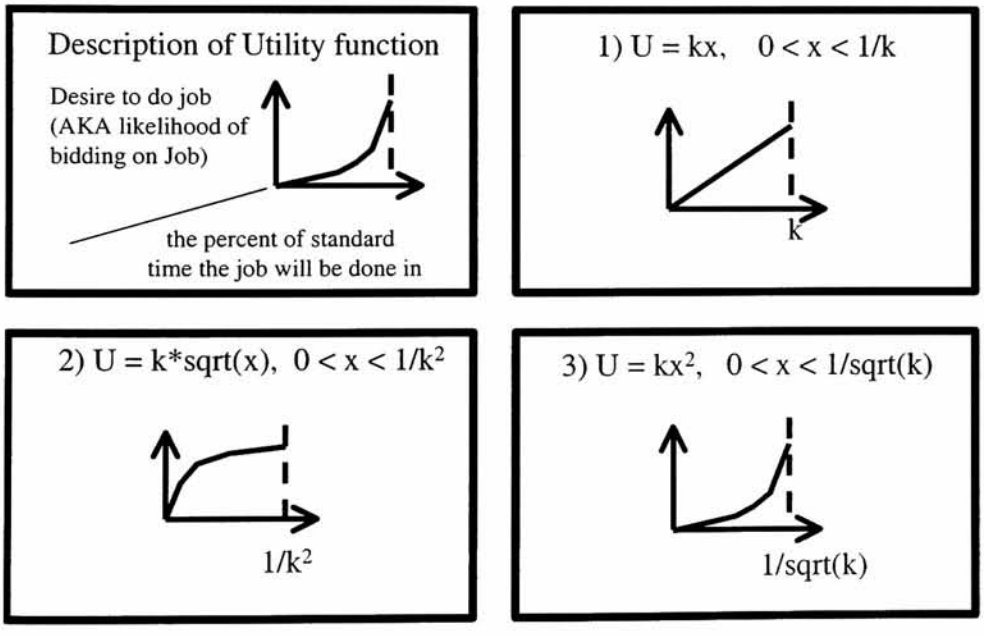


Figure 7-1 Utility Curves for the Assembler Agents

The Utility Curves are the guidelines used by Assembler Agents to choose work that is agreeable with them. By decreasing the likelihood of bidding at jobs Assembler Agents do not like, and increasing the likelihood of bidding on Jobs they do like, the Assembler Agents force their average utility upward.

Curve 1 represents an Assembler who is relatively impartial to the Work Area with a Job Agent requesting the bid. Thus, an Assembler Agent with Utility Curve 1 bids 100% of the time at 100% of standard, 90% of the time at 90% of standard and so on.

Assembler Agents identify WAs they would prefer to work at by assigning it a Utility Curve 2. An Assembler Agent with a Utility Curve 2 for a WA bids 100% of the time at 100% of standard, 95% of the time at 90% of standard, 92% of the time at 85% of standard and so on. Essentially they are more likely, relative to the other utility curves, to bid on a job they like even if the rate at which they must work increases.

Finally, assemblers apply a Utility Curve 3 to WAs for which they prefer not to work. An Assembler Agent with Utility Curve 3 for a WA bids 100% of the time at 100% of standard, 81% of the time at 90% of standard, and 72% of the time at 85% of standard and so on. In this case an Assembler Agent is less likely, relative to the other utility curves, to bid for a WA as the rate of production increases. An Assembler Agent Utility Curve chart for each agent at each WA can be found in Appendix B.

JOB AGENT

A unique Job Agent that represents each product as it is processed through the assembly line. Job Agents are local and physical agents as well. They represent a

physical element of the real system (a job as it is processed), and have limited interaction with other agents in the system (Shen and Norrie 11).

JOB AGENTS' GOAL AND MEANS OF ACHIEVING IT

The Goal of the Job Agent is to drive down the unit cost of each processing step by finding the lowest bidding Assembler Agent to complete the job. The Job Agent and Assembler Agents for a particular job at a particular WA use the bidding scheme documented in the next section to decide which Assembler Agent performs the work required. Through several rounds of bidding the Job Agent attempts to find an Assembler Agent that meets an "ideal" or "acceptable" bidding price. Appendix C documents the ideal and acceptable bid criteria for the Job Agent.

THE JOB AGENT AND ASSEMBLER AGENT BIDDING STRUCTURE

The Job Agent and Assembler Agent have goals related to individual production process steps. The Job Agent's goal is to complete each process step in the most cost effective manner, while the Assembler Agent's goal is to work on portions of the process that it finds most satisfying. These two goals are not congruent; in fact they are divergent. The Assembler Agents goal of maximizing average utility is correlated to the Assembler Agent producing the job at a higher percent of standard processing time. It is less likely to bid on the Jobs as the percent of standard processing time decreases. This is essentially the antithesis of the Job Agent's goal.

To reconcile the differences in the goals of these two agents, a negotiation structure represented by **Figure 7-2** is used. At each Work Area the Job Agent invokes

logic intended to decide which Assembler Agent to assign. This logic requests the following parameters: the starting percentage of standard (see the Supervisor Agent section of this chapter), the product type, the Sales Dollars associated with that product type, and the Work Area. These parameters are used by the attendant simulation subroutines to help determine which Assembler Agent to assign.

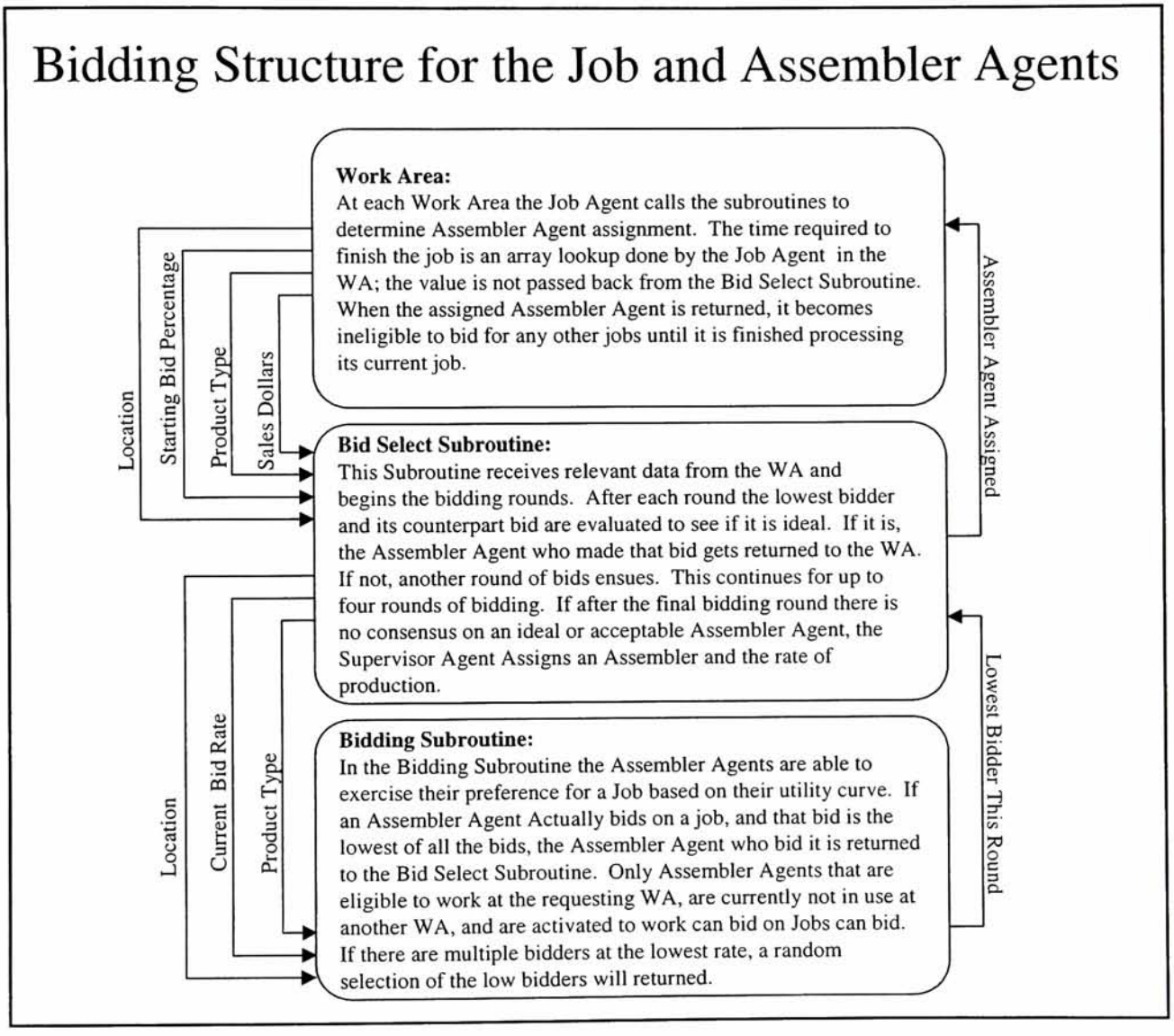


Figure 7-2 Job Agent - Assembler Agent Bidding Structure

The Bid Select Subroutine serves as an intermediary between the Job Agent and Assembler Agent by engaging a bid cycle that can iterate up to four times before returning an Assembler Agent assignment to the requesting Job Agent. The Bid Select Subroutine, request bids from Assembler Agents via the Bidding Subroutine, which identifies the lowest bidder for each round of bidding. The Job Agent, at this point, evaluates the lowest bid against and "ideal" criterion (detailed in Appendix C). If the bid is ideal the associated Assembler Agent is assigned to the job. If the low bid is not ideal, another round of bidding ensues.

Each round of bidding lowers the Assembler Agents' current bid rate by a set amount until the threshold of 85% of standard processing time (the lowest percent of standard an operator is willing to work) is reached or an ideal bid is received. If no ideal bids have been received after all eligible rounds of bidding have completed, the Job Agent seeks an "acceptable" bid. The criterion for acceptable bids is found in Appendix C. Assuming multiple acceptable bids, the lowest bidding Assembler Agent is assigned. If no acceptable bids are made, then the Supervisor Agent is activated assigning the eligible and active Assembler Agent with the highest average utility at the lowest percent of standard.

The Bidding Subroutine determines whether or not an Assembler Agent bids. The probability of an eligible, free, and active Assembler Agent bidding is based on each Assembler Agent's utility curve, as previously discussed (see also Appendix B). Only eligible Assembler Agents that are active and not currently in use can place a bid.

SUPERVISOR AGENT

The Supervisor Agent is a system level functional agent concerned with three tasks. Resource Allocation is the first task the Supervisor Agent undertakes. The function of Resource Allocation is to determine the rate at which Assembler Agents begin their bidding (either at 100% or at 90% of the standard processing time). Assigning Agents to jobs is the second task of a Supervisor Agent. The Supervisor Agent sets a rate and assigns an Assembler Agent to jobs when the Job Agent Assembler Agent bidding structure fails. Resource Activation and Deactivation is the third task of the Supervisor Agent. This task adds or removes Assembler Agents from the system.

SUPERVISOR AGENT'S GOAL AND MEANS OF ACHIEVING IT

The Supervisor Agent can be thought of as a single agent that undertakes multiple tasks, with the goal of the Supervisor Agent to Maximize Net Profit. The first means by which the Supervisor Agent attempts to achieve its goal is by minimizing operating expense and late penalty by optimizing resource allocation (starting percentage of standard processing time). This is referred to in **Figure 7-1** and later in the chapter as the "First Level of System Control". The "Second Level of System Control" the Supervisor Agent can enable is the resource assignment; this is explicitly instructing a specific Assembler Agent to do a specific job at a specific rate. Finally, via the "Third Level of System Control", the Supervisor Agent activates or deactivates assemblers based on the daily late fees and the number of assembler assignments made per day.

RESOURCE ALLOCATION

AGENT INTERACTION FOR RESOURCE ALLOCATION

The resource allocation of the Complex Adaptive System model is more dynamic than the Base Case and the Theory of Constraints. The four software agents that are new to the CAS model make all of the resource allocation decisions. These agents, described above, include a Scheduler Agent, Assembler Agents, Job Agents, and a Supervisor Agent. Unlike the TOC and BC models, the CAS has no static resources that are assigned to only one Work Area (**Table 7-2**). Negotiations between agents via direct communication or use of common System Measures decide which Assembler Agents are assigned to a job at any given WA. The Scheduler Agent, Assembler Agents, and Job Agents all work at achieving their own goals for the entire system run. The Supervisor agent is only enabled when needed. The three levels of system control described in the next section refer to the Supervisor Agent's functions.

	Static Resource Allocation	Dynamic Resource Allocation
Base Case	X	
Theory of Constraints	X	X
Complex Adaptive System		X

Table 7-2 The CAS Model Has Only Dynamic Resource Allocation

The First Level of System Control: Assembler Agent Allocation Based on Queue Rearrangement Costs

The Supervisor Agent and Scheduler Agent Communicate via the queue rearrangement cost, a common system metric. Every two hours, based on the previous two-hour total of queue rearrangement costs, the Supervisor Agent either requires Assembler Agents to begin bidding at 100% or 90% of the standard processing time for all jobs in the system. If the queue rearrangement cost is greater than an internal system cutoff determined by the Supervisor Agent, then the Assembler Agents are required to begin bidding at 90% of standard processing time for the next two hours. This is a proactive step taken by the Supervisor Agent. The increased occurrence of queue rearrangement is a prognostic indicator that the system may soon incur late penalties. By setting the maximum Assembler Agent bid rate to 90% of standard processing time the Supervisor Agent speeds up production slightly to help avoid potential late fees.

The Second Level of System Control: Assembler Agent Assignment

The Supervisor Agent is activated to assign Assembler Agents to Jobs when the Assembler Agent Job Agent bidding structure fails. The Supervisor assigns the Assembler Agent with the highest average utility that is also eligible and available to work at the requesting Work Area. It is assigned at 85 % of standard processing time, thus driving down the Assembler Agent's average utility.

THE THIRD LEVEL OF SYSTEM CONTROL: ACTIVATING OR DEACTIVATING ASSEMBLER AGENTS

The Supervisor Agent's third control level is to add or remove Assembler Agents. This decision is made based on the daily late fee incurred by the system and the number of times the Scheduler Agent is used to rearrange the Incoming Job Queue. The Complex Adaptive System scenarios (highly, moderately, and lightly stressed) all start with eight Assembler Agents consisting of two Alpha Agents, three Beta Agents and three Gamma Agents.

ADDING ASSEMBLER AGENTS

An Assembler Agent is added to the system if the late fee per day is greater than the average cost of one operator per day, **or** if the Supervisor Agent assigns assemblers more than once a day (based on the Supervisor Agent's second level of system control: Assembler Agent assignment). This makes adding an Assembler Agent relatively easy, because only one of these conditions must occur.

REMOVING ASSEMBLER AGENTS

An Assembler Agent can only be removed at the end of an eight-hour day. The Supervisor Agent removes an Assembler Agent if there is no late fee for the day **and** the Supervisor Agent is never used to assign assemblers (the Supervisor Agent's second level of system control). Thus, removing an Assembler Agent is more difficult than adding one, which is consistent with a real build-to-order manufacturing line set up with short

lead times. The system slightly favors having more capacity than it requires, at a slightly higher personnel cost, than having a shortage of capacity at a slightly lower personnel cost.

JUSTIFICATION FOR THE COMPLEX ADAPTIVE SYSTEM MODEL

The arrow diagram in **Figure 7-3** is a framework for the justification of the Complex Adaptive System model, which includes several new elements. The Assembler, Scheduler, Job, and Supervisor Agents are the most significant new editions. Unlike the Base Case and Theory of Constraints models there are instances of looping in the CAS model.

The looping in the CAS is created by the interaction between the different agents. For example a loop is created between the Assembler, Job and Supervisor Agents to assign Assembler Agents when required. If the unit labor cost of the Assembler Agents is too high, for each respective bid, the Job Agent requests the Supervisor Agent to assign an Assembler Agent.

A table of definitions follows the Arrow Diagram for added assistance in reading (**Table 7-3**). The From-To Relationships, **Table 7-4**, follows the Arrow Diagram and details the relationship between each element of **Figure 7-3**. Other details of the arrow diagram are discussed further in the allocation of resources section of this chapter. Recall that the competition between positive and negative loops and the rates involved in each loop determine the dynamics of the system.

Each agent has a unique goal and a means of achieving it. This is in stark contrast to the Theory of Constraints where all elements of the system strive to achieve one common goal.

SYSTEM MEASURES

As with the Base Case model and the Theory of Constraints model, the arrow diagram (**Figure 7-3**) gives insights into some of the interactions between the model and measures of the system. All of the elements of the Net Profit equation are now applicable and all System Measures are enabled. Queue rearrangement is a function of the Scheduling Agent, and a cost for scheduling is charged for each time the queue is altered.

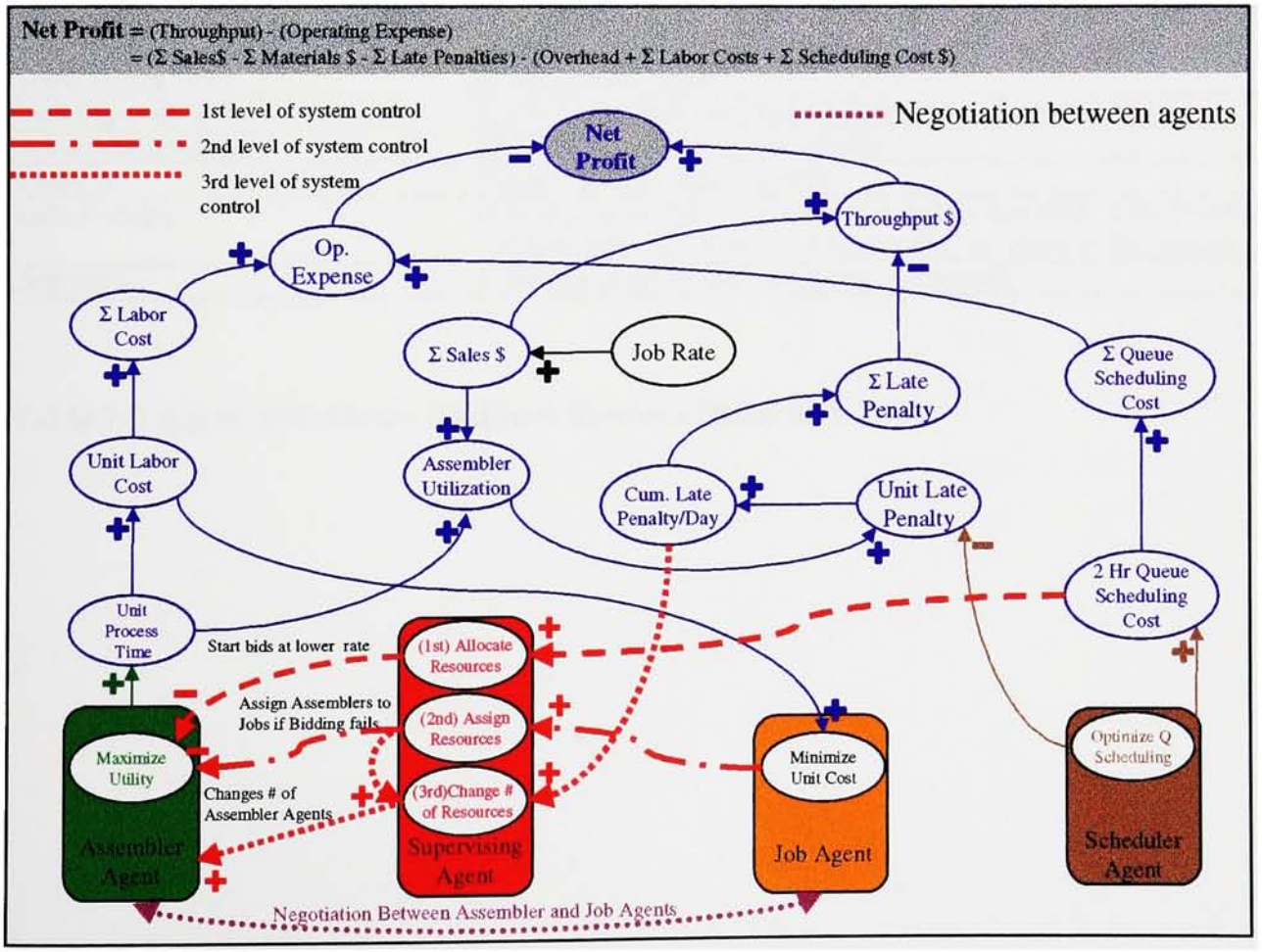


Figure 7-3 The Complex Adaptive System Arrow Diagram

Agents and System Elements	Description
Net Profit = (Throughput) - (operating expense)	= (Sum Sales$ - Sum Materials $ - Sum Late Penalties) - (Overhead $+ Sum Labor $+ Sum Expediting $)
Scheduling Agent	Rearranges the Queue based on sales dollars each product represents. High Sales Dollar Products get processed sooner, to help reduce the late fees (More expensive products will have higher late fees).
Assembler Agent	Attempts to maximize the assembler satisfaction based on their utility functions per station, bidding more frequently at jobs they prefer.
Job Agent	Drives down Unit Cost by finding the lowest bidding Assembler Agent to do the job
Supervisor Agent	Sets Assembler Agent Bid level at 100% or 90% based on the performance of the system. Assigns agents to do jobs when the Job Agent and Assembler Agents' bidding fail. Adds or Removes Resources when deemed necessary.
Net Profit	Global fitness measure of our system. Net Profit = Throughput – Operating Expense
Operating Expense	The measure used in The Net Profit Equation. Operating Expense = (Overhead + Sum of Labor + Sum of Scheduling Costs)
Inventory	Inventory is considered any money that gets spent on purchasing things the company intends to sell
Overhead	Any expense related to the cost of turning Inventory into Throughput (Includes Unproductive Labor)
Throughput	A measure used in the Net Profit Equation. Throughput = (Sum of $ from Sales –Sum of materials costs – Sum of late penalties)
Total Labor Cost	Sum of labor costs
Unit Process Time	Time to produce one product or one step of one product
Scheduling Costs	Activity based cost of Scheduling
Sales $	Sum of $ from Sales of units.
Late Penalty	Penalty assessed if job not completed by due date. Late Penalty = 10% of Sales price per product per day, prorated to the minute.
Job Rate	The rate at which jobs come into the system

Table 7-3 Agent and Arrow Diagram Element Definition Table

CAS		
From	To	Description
Job Rate	Sum of Sales Dollars	As more jobs enter the system there will be a greater potential for increased sales revenue.
Assembler Agent	Unit Process Time	As the utility of an assembler agent is increased (from an assembler agent perspective), unit process time will also increase.
Assembler Agent	Job Agent	The Assembler Agent and Job Agent Negotiate based on local goals to decide which assembler will process which job at a given Work Area.
Scheduling Agent	Unit Late Penalty	As jobs in the queue are rearranged, it is expected that the per unit late penalty will decrease.
Scheduling Agent	2 Hr Scheduling Cost	Rearranging jobs in the queue will increase the per unit scheduling cost of jobs in the system.
Supervisor Agent	Assembler Agent	Supervisor Agent Assignment and Allocation of assemblers lowers the average utility of the Assembler Agents.
Job Agent	Supervisor Agent	When the bidding fails between the Assembler and Job Agents, the Supervisor is activated to assign an Assembler Agent to the Job
Assembler Utilization	Unit Late Penalty	As the utilization of the Assembler Agents increases past the capacity of the system, the late penalty incurred for the jobs increases.
Unit Late Penalty	Sum of Daily Late Penalties	As each unit late penalty increases, the total late penalties for the day also increases.
Sum of Daily Late Penalties	Sum of Late Penalties	As the daily late penalty increases, the sum of late penalties also increases.
Sum of Late Penalties	Throughput	As the overall late penalty increases, the system throughput t(Dollars) decreases.
Sum of Daily Late Penalties	Supervisor Agent	As the total late penalty costs per day increase above a threshold value, the supervisor is requested to increase the number of resources
Supervisor Agent	Supervisor Agent	As the Supervisor Agent Assigns multiple Assembler Agents to jobs per day, the number of Resources in the system is increased
Sales Dollars	Throughput	The more Sales Dollars generated, the more Throughput generated based on the Net Profit equation.
Sales Dollars	Assembler Utilization	As more sales are accumulated, assembler utilization must increase in order to accommodate the increased demand.
Throughput	Net Profit	Increasing throughput increases net profit.
2 Hr Scheduling Cost	Sum of Scheduling Costs	As the scheduling costs for each 2hr time period increase, the total scheduling costs for all jobs will also increase.
Sum of Scheduling Costs	Operating Expense	Higher total scheduling costs results in a higher operating expense for the system.
Operational Expense	Net Profit	Increasing operating expenses decreases net profit (See Net Profit equation)
Unit Process Time	Unit Labor Cost	As the processing time for each unit increases, the labor cost associated with producing that unit also increases.
Unit Process Time	Assembler Utilization	As the processing time for each unit increases, the utilization of the assembler agents also increases to accommodate the change.
Unit Labor Cost	Sum of Labor Costs	As the cost associated with the labor allocated to each unit increases, the total labor cost also increases.
Sum of Labor Costs	Operating Expense	As the total labor cost increases, the operating expense of the system also increases.

Table 7-4 From To Description of the CAS Arrow Diagram

ADDITIONAL ASSUMPTIONS OF THE COMPLEX ADAPTIVE SYSTEM

Beyond the generic assumptions for all the models, the CAS model also assumes:

- Assembler Agents can opt to not bid on a job.
- Alpha Assembler Agents are able to work at any WA.
- Beta Assembler Agents are only able to work at Work Areas 1-5.
- Gamma Assembler Agents are only able to work at Work Areas 6-10.

CHAPTER 8 SIMULATION RESULTS

SYSTEM MEASURES RESULTS

The three sets of System Measures results (**Tables 8-1**, **8-2**, and **8-3**) detail the system-by-system output for the measures first described in Chapter 4. The fast arrival scenarios (highly stressed systems) were each replicated three times. Recall that the highly stressed systems show the most potential for improvement, and are of the most interest to this research. Multiple replication allows for statistical analysis among model types (CAS, TOC, and BC).

The medium arrival paced (moderately stressed) and slow arrival paced (lightly stressed) systems were run once for each of the manufacturing theories employed. These stress levels of the system, being of less interest to the system, did not warrant multiple replications.

NET PROFIT			
	BC FAST	TOC FAST	CAS FAST
replication 1	$ 66,650.97	$134,008.03	$ 168,934.13
replication 2	$ 66,661.36	$141,505.20	$ 167,618.35
replication 3	$ 66,632.14	$134,152.81	$ 176,080.00
AMOUNT OF POTENTIAL SALES DOLLARS			
	BC FAST	TOC FAST	CAS FAST
replication 1	$139,250.00	$208,500.00	$ 209,350.00
replication 2	$139,250.00	$208,500.00	$ 210,500.00
replication 3	$139,250.00	$208,850.00	$ 209,350.00
LATE PENALTIES			
	BC FAST	TOC FAST	CAS FAST
replication 1	$ 39,479.03	$ 8,571.97	$ 3,115.87
replication 2	$ 39,468.64	$ 5,074.80	$ 1,721.65
replication 3	$ 39,497.86	$ 8,777.19	$ -
TOTAL THROUGHPUT			
	BC FAST	TOC FAST	CAS FAST
replication 1	$ 99,770.97	$199,928.03	$ 206,234.13
replication 2	$ 99,781.36	$203,425.20	$ 208,778.35
replication 3	$ 99,752.14	$200,072.81	$ 209,350.00
TOTAL RESOURCE COSTS			
	BC FAST	TOC FAST	CAS FAST
replication 1	$ 33,120.00	$ 65,920.00	$ 36,320.00
replication 2	$ 33,120.00	$ 61,920.00	$ 40,320.00
replication 3	$ 33,120.00	$ 65,920.00	$ 33,120.00

Table 8-1

NET PROFIT			
	BC MEDIUM	TOC MEDIUM	CAS MEDIUM
replication 1	$ 99,189.22	$ 98,472.36	$ 122,948.03
replication 2			
replication 3			
AMOUNT OF POTENTIAL SALES DOLLARS			
	BC MEDIUM	TOC MEDIUM	CAS MEDIUM
replication 1	$139,250.00	$148,400.00	$ 150,250.00
replication 2			
replication 3			
LATE PENALTIES			
	BC MEDIUM	TOC MEDIUM	CAS MEDIUM
replication 1	$ 6,940.78	$ 7.64	$ 791.97
replication 2			
replication 3			
TOTAL THROUGHPUT			
	BC MEDIUM	TOC MEDIUM	CAS MEDIUM
replication 1	$132,309.22	$148,392.36	$ 149,458.03
replication 2			
replication 3			
TOTAL RESOURCE COSTS			
	BC MEDIUM	TOC MEDIUM	CAS MEDIUM
replication 1	$ 33,120.00	$ 49,920.00	$ 26,240.00
replication 2			
replication 3			

Table 8-2

NET PROFIT			
	BC SLOW	TOC SLOW	CAS SLOW
replication 1	$ 82,080.00	$ 73,780.00	$ 79,231.58
replication 2			
replication 3			
AMOUNT OF POTENTIAL SALES DOLLARS			
	BC SLOW	TOC SLOW	CAS SLOW
replication 1	$115,200.00	$115,700.00	$ 110,300.00
replication 2			
replication 3			
LATE PENALTIES			
	BC SLOW	TOC SLOW	CAS SLOW
replication 1	$ -	$ -	$ 4,408.42
replication 2			
replication 3			
TOTAL THROUGHPUT			
	BC SLOW	TOC SLOW	CAS SLOW
replication 1	$115,200.00	$115,700.00	$ 105,891.58
replication 2			
replication 3			
TOTAL RESOURCE COSTS			
	BC SLOW	TOC SLOW	CAS SLOW
replication 1	$ 33,120.00	$ 41,920.00	$ 26,240.00
replication 2			
replication 3			

Table 8-3

Table 8-1 System Measures for Fast (Highly Stressed) Models

Table 8-2 System Measures for Medium (Moderately Stressed) Models

Table 8-3 System Measures for Slow (Lightly Stressed) Models

COMPLEXITY (POWER LAW) RESULTS

POWER LAW RESULTS - HIGHLY STRESSED SYSTEM

The following is a representative sampling of the results of the Power Law (Complexity) Measures of interest. The measures are as follows: **Figure 8-1** Late Fee Per Late Product, **Figure 8-2** Percentage of Actual Sales For Late Jobs, **Figure 8-3** Time Between Late Exits, **Figure 8-4** Number of Late Jobs Per Day, **Figure 8-5** Throughput Per Day, and **Figure 8-6** Maximum Queue Size Per Day. Each one of the examples is from the highly stressed first replication of the Complex Adaptive System. Appendix A includes all three replications of the Power Law plots for the highly stressed cases, as well as one replication for each measure of the moderately and lightly stressed systems.

Recall, from Chapter 2, that the plots with a good linear fit display Self-Organized Critical (Complex) behavior. The x-axis in the plots represents a magnitude or time measurement, and the y-axis represents the cumulative probability that magnitude or time measurement.

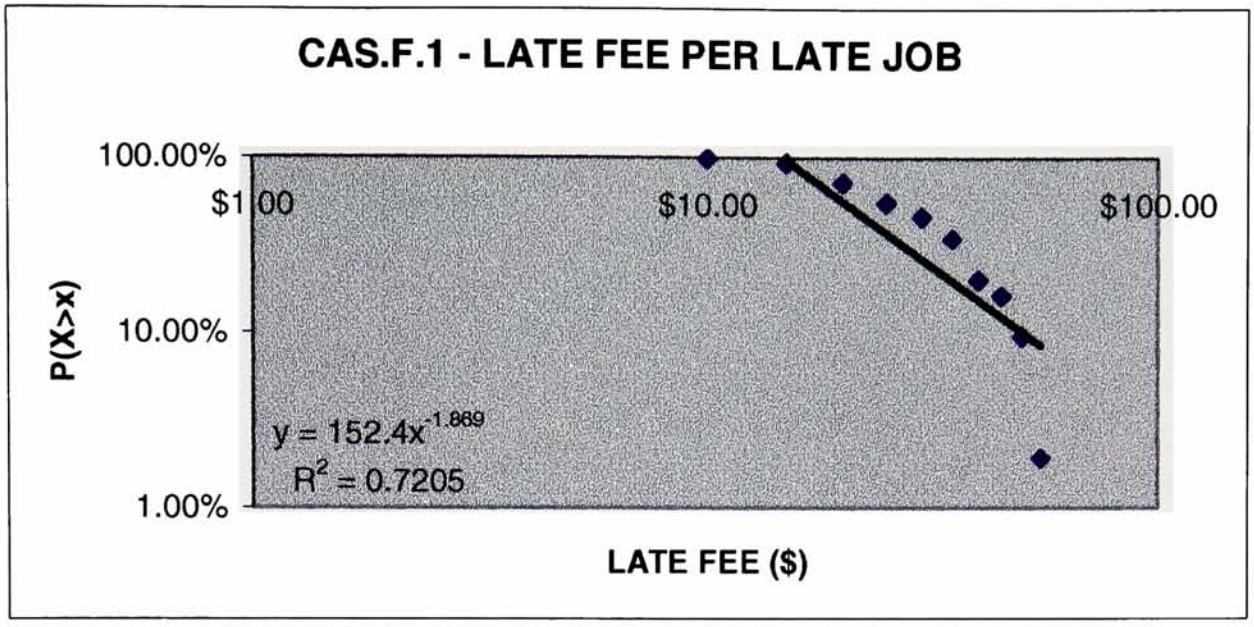


Figure 8-1 Late Fee Per Late Job

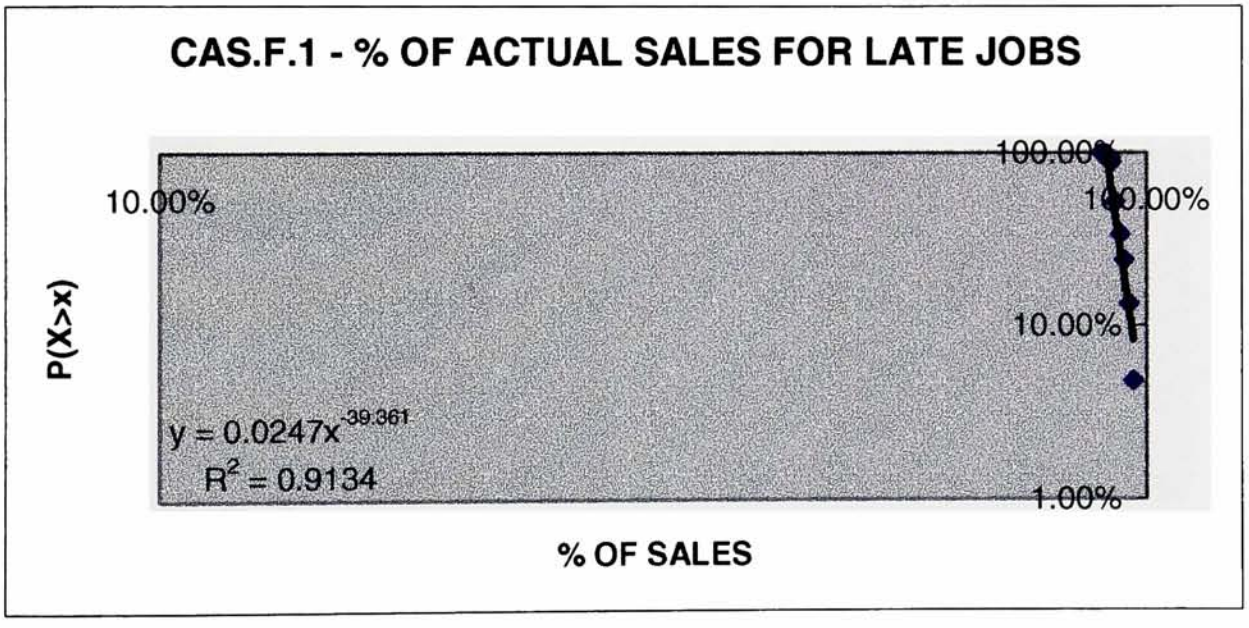


Figure 8-2 Percentage of Actual Sales for Late Jobs

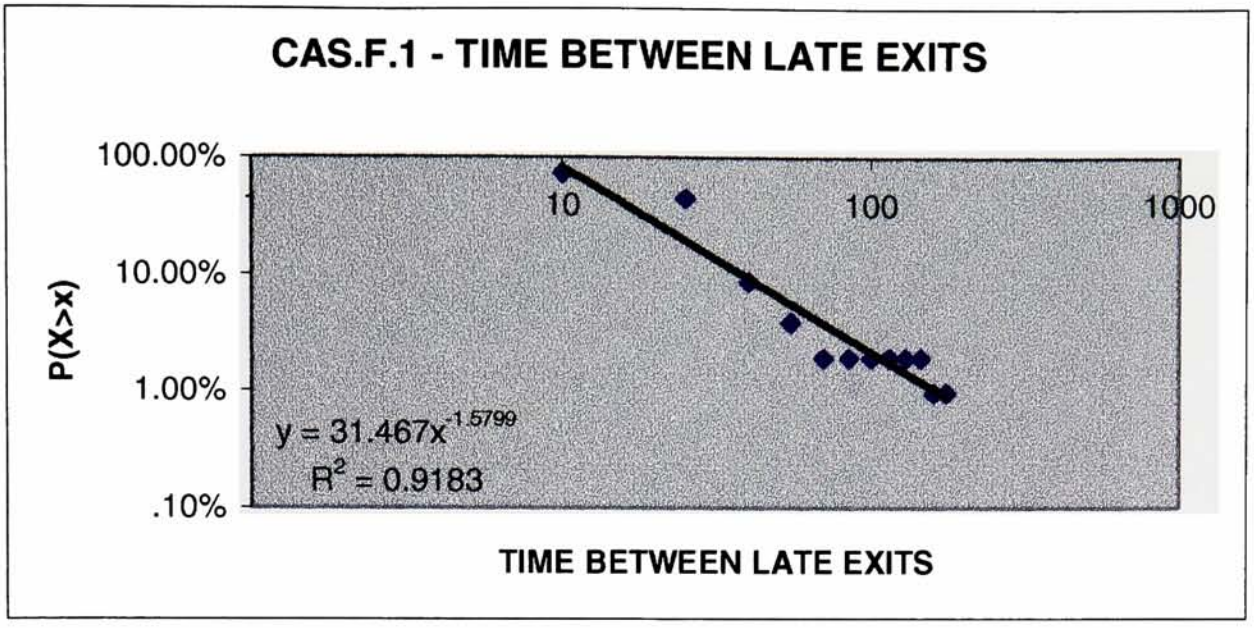


Figure 8-3 Time Between Late Exits

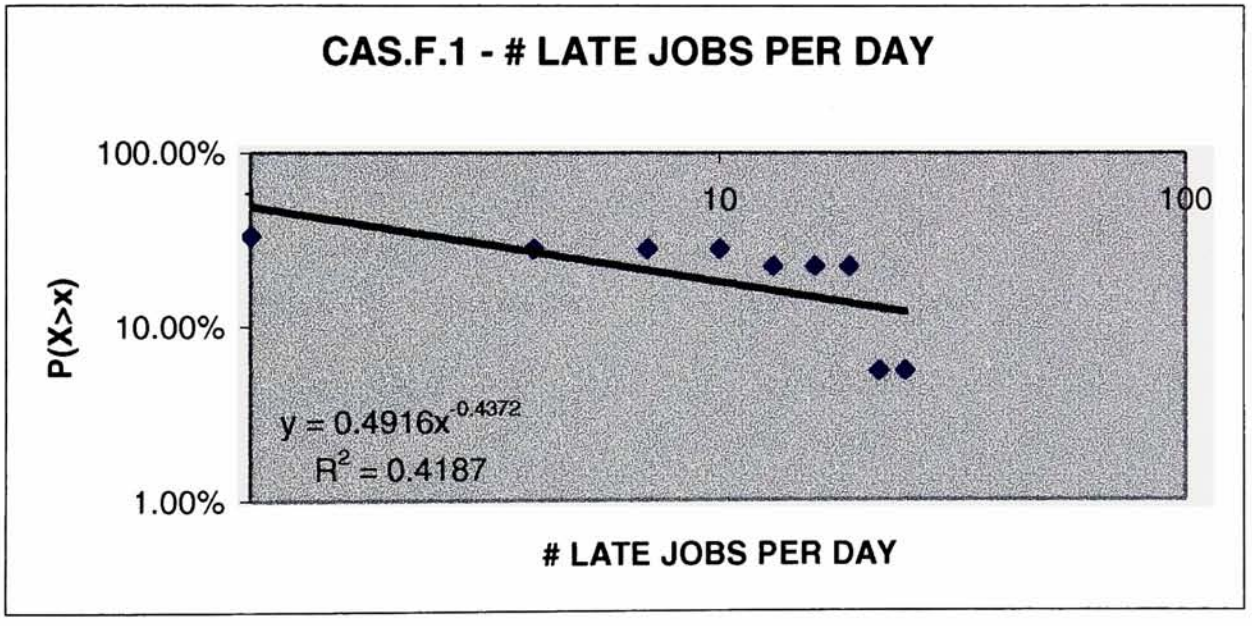


Figure 8-4 Number of Late Jobs Per day

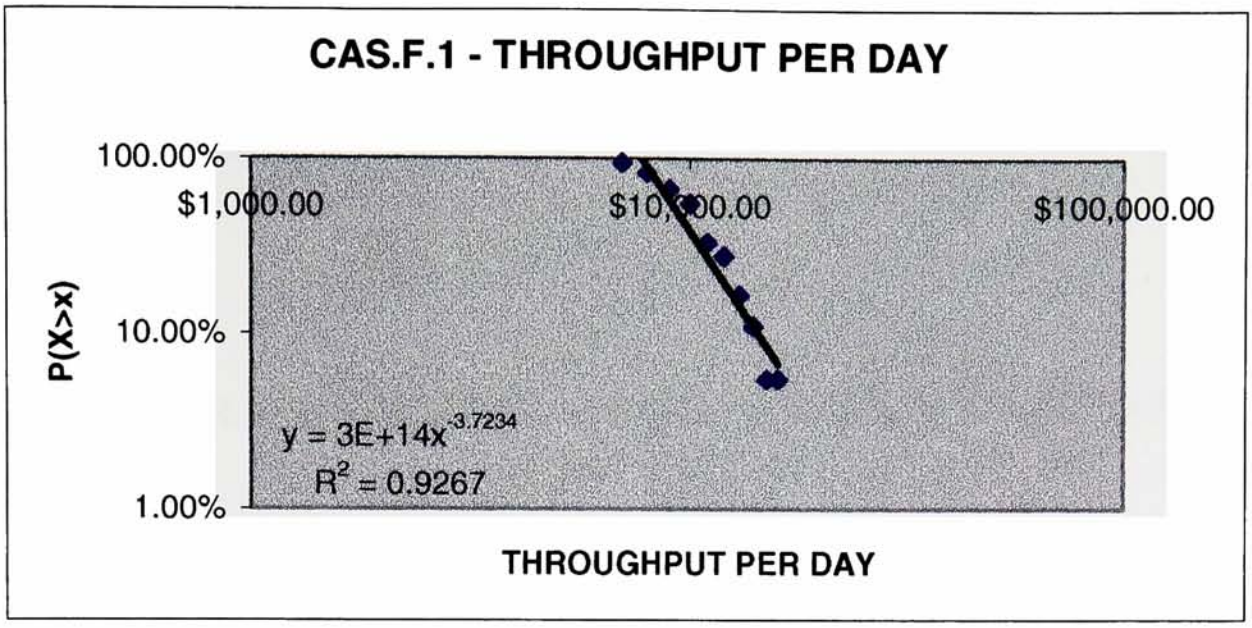


Figure 8-5 Throughput Per Day

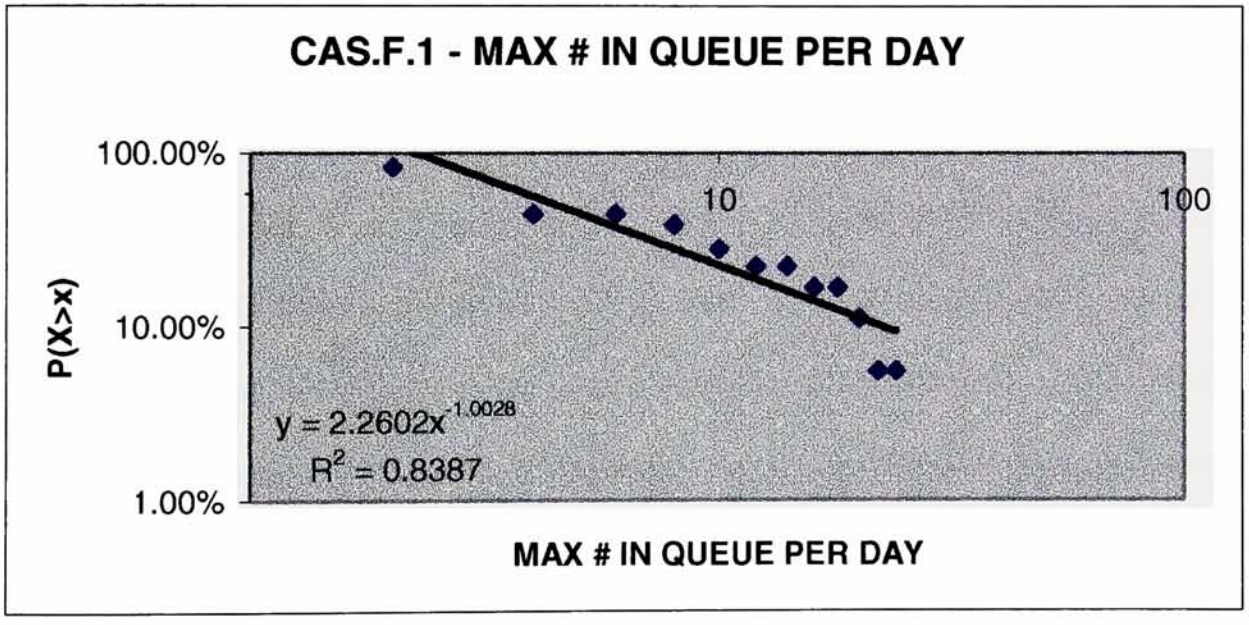


Figure 8-6 Maximum Number in the Queue Per Day

CHAPTER 9 ANALYSIS AND DISCUSSION OF RESULTS

DUNCAN'S MULTIPLE RANGE TEST FOR SYSTEM TO SYSTEM COMPARISON

In order to state with a degree of certainty that some models in this thesis out perform others, a statistical test is performed. Duncan's Multiple Range Test (MRT) requires samples of equal size and compares the means of each sample for significant difference (Montgomery 103, 675). This test is used because it is effective at choosing differences in means when true differences exist (Montgomery 105). There are three treatment levels (a), and three samples per treatment (n) for a total of 9 observations (N), and six degrees of freedom (f) for the mean square error. The test were conducted with a test error level of $a = 0.05$. The following are the results of five System Measures of interest. They include Net Profit Table 9-1, Amount of Potential Sales Dollars Table 9-2, Late Penalty Table 9-3, Total Throughput Table 9-4, and Total Resource Cost Table 9-5.

NET PROFIT	BC FAST	TOC FAST	CAS FAST
replication 1	$ 66,650.97	$ 134,008.03	$ 168,934.13
replication 2	$ 66,661.36	$ 141,505.20	$ 167,618.35
replication 3	$ 66,632.14	$ 134,152.81	$ 176,080.00
AVE	$ 66,648.16	$ 136,555.35	$ 170,877.49
MSE		$ 13,037,882.48	
STD ERROR OFEACH MEAN		$ 2,084.70	
BC AVE	$ 66,648.16	R2 =	$ 7,213.05
TOC AVE	$ 136,555.35	R3 =	$ 7,463.21
CAS AVE	$ 170,877.49		
			SIGNIFICANT?
CAS VS BC	$ 104,229.34	> $7,213.05	YES
CAS VS TOC	$ 34,322.15	> $7,463.21	YES
TOC VS BC	$ 69,907.19	> $7,463.21	YES

Table 9-1

AMOUNT OF POTENTIAL SALES DOLLARS	BC FAST	TOC FAST	CAS FAST
replication 1	$ 139,250.00	$ 208,500.00	$ 209,350.00
replication 2	$ 139,250.00	$ 208,500.00	$ 210,500.00
replication 3	$ 139,250.00	$ 208,850.00	$ 209,350.00
AVE	$ 139,250.00	$ 208,616.67	$ 209,733.33
MSE		$ 160,555.56	
STD ERROR OFEACH MEAN		$ 231.34	
BC AVE	$ 139,250.00	R2 =	$ 800.44
TOC AVE	$ 208,616.67	R3 =	$ 828.20
CAS AVE	$ 209,733.33		
			SIGNIFICANT?
CAS VS BC	$ 70,483.33	> $ 828.20	YES
CAS VS TOC	$ 1,116.67	> $ 800.44	YES
TOC VS BC	$ 69,366.67	> $ 800.44	YES

Table 9-2

LATE PENALTIES	BC FAST	TOC FAST	CAS FAST
replication 1	$ 39,479.03	$ 8,571.97	$ 3,115.87
replication 2	$ 39,468.64	$ 5,074.80	$ 1,721.65
replication 3	$ 39,497.86	$ 8,777.19	$ -
AVE	$ 39,481.84	$ 7,474.65	$ 2,418.76
MSE		$ 2,255,440.02	
STD ERROR OFEACH MEAN		$ 867.07	
CAS AVE	$ 2,418.76	R2 =	$ 3,000.07
TOC AVE	$ 7,474.65	R3 =	$ 3,104.12
BC AVE	$ 39,481.84		
			SIGNIFICANT?
BC VS CAS	$ 37,063.08	> $3,000.07	YES
TOC VS CAS	$ 5,055.89	> $3,104.12	YES
BC VS TOC	$ 32,007.19	> $3,104.12	YES

Table 9-3

TOTAL THROUGHPUT	BC FAST	TOC FAST	CAS FAST
replication 1	$ 99,770.97	$ 199,928.03	$ 206,234.13
replication 2	$ 99,781.36	$ 203,425.20	$ 208,778.35
replication 3	$ 99,752.14	$ 200,072.81	$ 209,350.00
AVE	$ 99,768.16	$ 201,142.01	$ 208,120.83
MSE		$ 2,222,192.92	
STD ERROR OFEACH MEAN		$ 860.66	
BC AVE	$ 99,768.16	R2 =	$ 2,977.87
TOC AVE	$ 201,142.01	R3 =	$ 3,081.15
CAS AVE	$ 208,120.83		
			SIGNIFICANT?
CAS VS BC	$ 108,352.67	> $ 2,977.87	YES
CAS VS TOC	$ 6,978.81	> $3,081.15	YES
TOC VS BC	$ 101,373.86	> $3,081.15	YES

Table 9-4

TOTAL RESOURCE COSTS	BC FAST	TOC FAST	CAS FAST
replication 1	$ 33,120.00	$ 65,920.00	$ 36,320.00
replication 2	$ 33,120.00	$ 61,920.00	$ 40,320.00
replication 3	$ 33,120.00	$ 65,920.00	$ 33,120.00
AVE	$ 33,120.00	$ 64,586.67	$ 36,586.67
MSE		$ 6,115,555.56	
STD ERROR OFEACH MEAN		$ 1,427.77	
BC AVE	$ 33,120.00	R2 =	$ 4,940.07
CAS AVE	$ 36,586.67	R3 =	$ 5,111.41
TOC AVE	$ 64,586.67		
			SIGNIFICANT?
TOC VS BC	$ 31,466.67	> $4,940.07	YES
TOC VS CAS	$ 28,000.00	> $5,111.41	YES
CAS VS BC	$ 3,466.67	< $5,111.41	NO

Table 9-5

Table 9-1 Duncan's Multiple Range Test for Net Profit

Table 9-2 Duncan's Multiple Range Test for Potential Sales Dollars

Table 9-3 Duncan's Multiple Range Test for Late Penalties

Table 9-4 Duncan's Multiple Range Test for Total Throughput

Table 9-5 Duncan's Multiple Range Test for Total Resource Cost

System Measures Discussion - Answering Research Question 1

Tables 9-1 to 9-5 indicate that the Complex Adaptive Systems significantly outperform both the Base Case and the Theory of Constraints systems, and the Theory of Constraints also outperforms the Base Case. In fact, for each System Measure except for Total Resource Cost, the Duncan's Multiple Range Tests show the same pattern; the CAS systems significantly outperform the TOC and the BC systems, and the TOC systems also outperform the BC systems.

Therefore, it is safe to say that the Complex Adaptive System as a whole produces more Net Profit, captures more of the Potential Amount of Sales (which essentially means it produces more product), and produces more Throughput, while incurring less Late Penalties and Resource Costs than the Theory of Constraints. For the Total Resource Cost Measure, the Complex Adaptive System and the Base Case both outperform the Theory of Constraints, but are not significantly different from one another. The Theory of Constraints likewise outperforms the Base Case in the same manner, with the exception of the Total Resource Cost measure where the Base Case is better.

Power Law Test for (Complex) Behavior

As stated in Chapters 1 and 2, the Power Law behavior may be an indicator for systems that are Complex, or in a Wolfram Class IV range (see Chapter 1). Three replications were run for each model in the highly stressed (fast arrival pace) scenarios.

Examples of the individual Power Law plots can be seen in Chapter 8 and a complete listing can be found in the Appendix A.

The following are graphs that trace the progression of the power and the fit of each individual plot. Research has shown that Self-Organized Critical systems tend to exhibit powers in the range of 1 2 (Bak "Complexity, Contingency, and Criticality" 6693; Bak "Self-Organized Criticality: A Holistic View of Nature" 449). Therefore, as the Power Law measures are plotted, it is of interest to this thesis if the powers converge the1-2 range described above, with a good or very good statistical fit to the equation generated as the models are evaluated from BC, to TOC, to CAS. A good statistical fit is considered any R^2 value greater that 0.8, and very good fit any R^2 fit greater than 0.9. The plots of R^2 vs. Power for each measure in the highly stressed cases are as follows:

LATE FEE PER LATE JOB

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	1.257	0.607		
BC.F.2	1.257	0.607	1.257	0.607
BC.F.3	1.257	0.607		
TOC.F.1	1.863	0.957		
TOC.F.2	1.563	0.856	1.656	0.869
TOC.F.3	1.571	0.869		
CAS.F.1	1.869	0.721		
CAS.F.2	1.992	0.816	1.931	0.768
CAS.F.3				

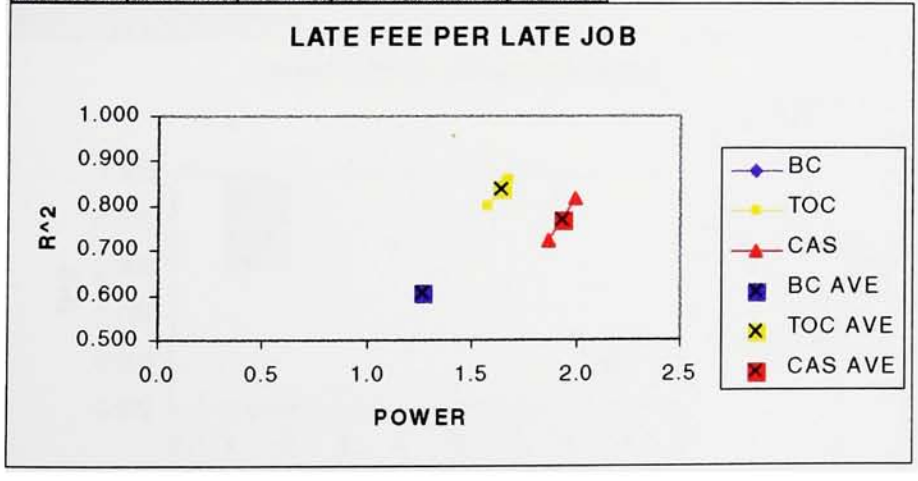


Figure 9-6 Late Fee Per Late Job Power Law Graph

% OF ACTUAL SALES PER LATE JOB

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	3.399	0.872		
BC.F.2	3.438	0.856	3.425	0.862
BC.F.3	3.438	0.856		
TOC.F.1	8.968	0.810		
TOC.F.2	26.018	0.827	19.418	0.800
TOC.F.3	16.270	0.761		
CAS.F.1	39.361	0.913		
CAS.F.2	39.097	0.828	39.229	0.871
CAS.F.3				

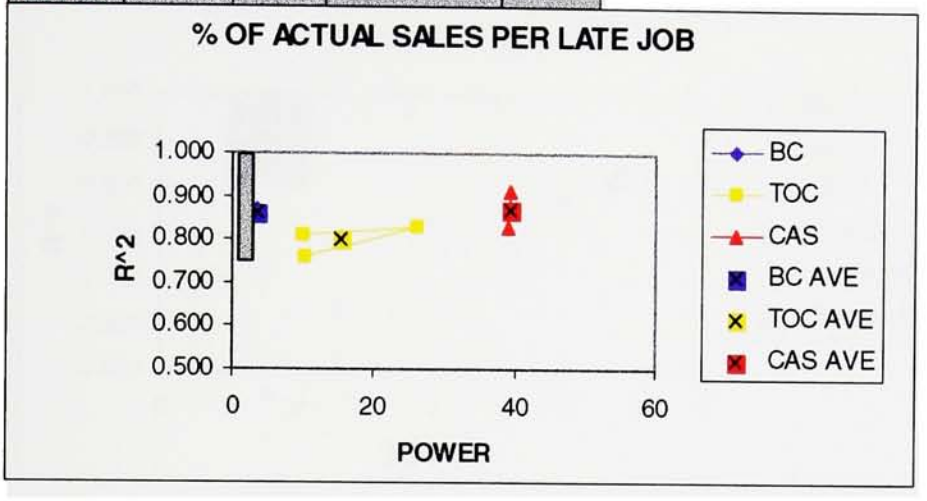


Figure 9-7 Percent of Actual Sales Per Late Job Power Law Graph

TIME BETWEEN LATE EXITS

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	7.173	0.498		
BC.F.2	7.990	0.663	7.781	0.607
BC.F.3	8.180	0.661		
TOC.F.1	1.511	0.629		
TOC.F.2	1.509	0.646	1.573	0.662
TOC.F.3	1.698	0.673		
CAS.F.1	1.580	0.918		
CAS.F.2	1.184	0.930	1.382	0.924
CAS.F.3				

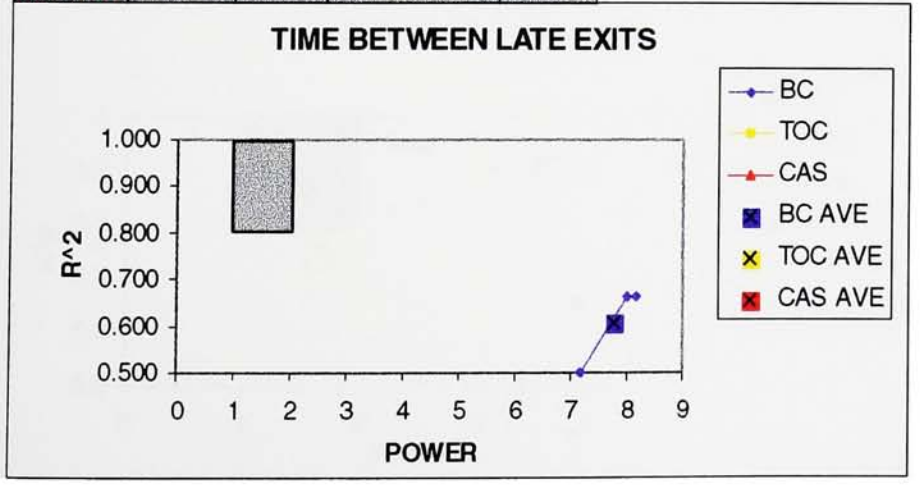


Figure 9-8 Time Between Late Exits Power Law Graph

LATE JOBS PER DAY

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	5.8922	0.780		
BC.F.2	7.0526	0.859	6.725	0.821
BC.F.3	7.2297	0.825		
TOC.F.1	1.331	0.660		
TOC.F.2	0.6977	0.417	0.998	0.591
TOC.F.3	0.9642	0.697		
CAS.F.1	0.4372	0.419		
CAS.F.2	0.6484	0.769	0.543	0.594
CAS.F.3				

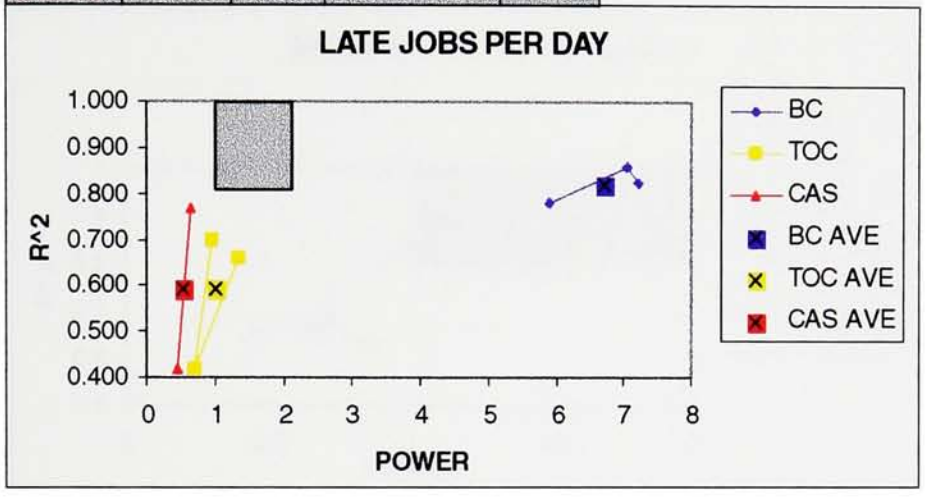


Figure 9-9 Number of Later Jobs Per Day

THROUGHPUT PER DAY

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	2.3325	0.9422		
BC.F.2	2.6152	0.952	2.52	0.95
BC.F.3	2.6152	0.952		
TOC.F.1	2.0249	0.7363		
TOC.F.2	1.6907	0.7593	1.76	0.74
TOC.F.3	1.5695	0.7256		
CAS.F.1	3.7234	0.9267		
CAS.F.2	3.1096	0.9059	3.57	0.91
CAS.F.3	3.8736	0.8965		

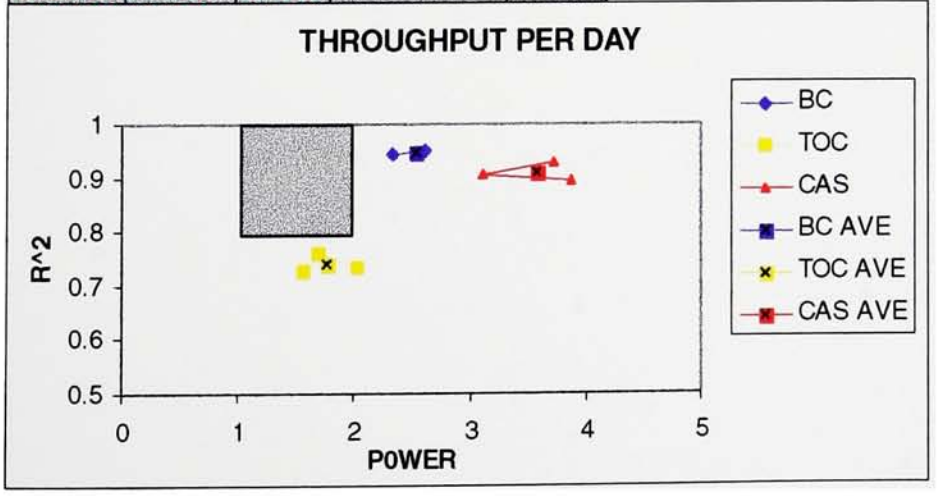


Figure 9-10 Throughput Per Day Power Law Graph

MAX # IN QUEUE PER DAY

MODEL	POWER	R^2	AVE. POWER	AVE R^2
BC.F.1	0.4551	0.6522		
BC.F.2	0.7009	0.6914	0.66	0.66
BC.F.3	0.8328	0.6274		
TOC.F.1	0.6416	0.675		
TOC.F.2	0.8937	0.7512	0.87	0.64
TOC.F.3	0.9837	0.5805		
CAS.F.1	1.0028	0.8387		
CAS.F.2	0.7262	0.7402	1.03	0.84
CAS.F.3	1.3524	0.9554		

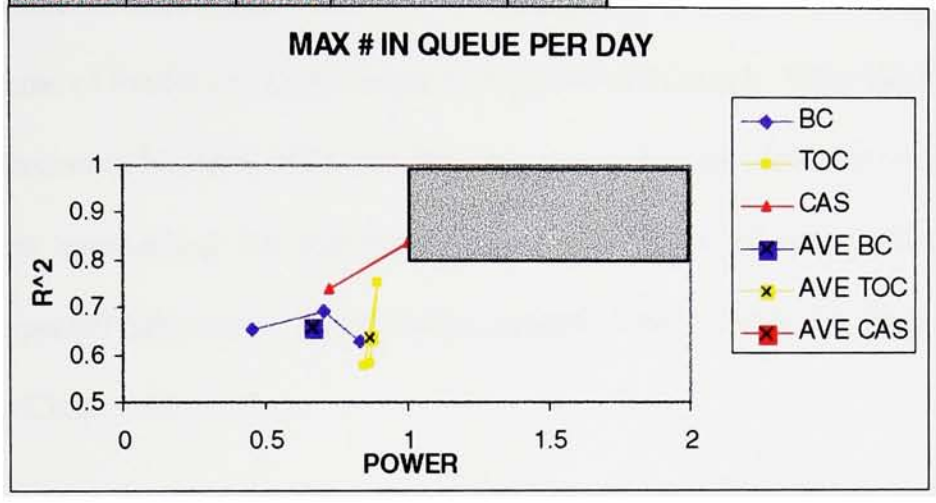


Figure 9-11 Maximum Number in Queue Per Day Power Law Graph

Discussion of Power Law (Complexity) Measures - Answering Research Question 2

There is no conclusive answer to research question two. A trend of increasing R^2 fits, and power measures converging on the 1 to 2 range is expected as the models are evaluated from Base Case, to Theory of Constraints, to Complex Adaptive System. Some of the Power Law measures displayed this trend. Time Between Late Exits and the Maximum Number in Queue Per Day measures both show increasing R^2 fits and Powers that approached the expected range. The other measures show at most one of the expected behaviors. These results suggest some avenues for future research as discussed in Chapter 10.

CHAPTER 10 AVENUES FOR FUTURE RESEARCH

Although the Complex Adaptive System model did significantly out perform both the Theory of Constraints and the Base Case models on almost all of the System Measures, the Power Law analysis of the output was inconclusive. The design of the CAS model was intended to create Complex or Self-Organized Critical Behavior. Some of the Power Law Measures studied showed a progression that lead toward what may be construed as an indicator of complexity, as the models advance from the BC to TOC to the CAS. Some of the other measures, however, did not. The following sections address some issues that may be implemented in future research to elicit more conclusive complex behavior.

ADD MORE VARIABILITY TO THE ARRIVAL FILE

Although the arrival files used in the CAS, TOC, and BC models were somewhat randomized for the type of product arriving and the quantity per arrival, some elements were arbitrarily rigid and patterned. The orders in the fast paced system arrived every sixty minutes, in the medium paced system every eighty-one minutes, and in the slow paced system every one hundred and two minutes. In each case, orders for products

arrived at equally spaced time events. Adding variability to the arrivals times (via an exponential time distribution) would more closely represent the activities in the natural world where complexity is found in abundance. By eliminating a patterned time arrival, the systems would no longer be driven by a structured drumbeat of incoming work. Complex systems exist in a very narrow band of parameters, and the rigid incoming pattern of jobs might be outside these parameters. This might be enough to nudge the CAS system out of a Class II into a Class IV behavior in terms of Wolfram's complexity classifications.

Sensitivity Analysis

Additional attention to some internal model characteristics in the CAS may yield improvement thorough sensitivity analysis. The internal measures of interest that may be fine tuned include system parameters such as the queue rearrangement cost, the cutoffs the Supervisor Agent uses to determine whether to add or remove resources (see Chapter 7 and Appendix C), and the percent of standard processing time at which the Assembler Agents are able to work. Ideal and acceptable levels for bids offer another opportunity for improvement, as does increasing sample size from which to draw conclusions.

Controlled Experiments Based on Arrow Diagram Loops

The Arrow Diagrams are a powerful tool used to identify interactions within a system as well as control loops, which optimize the system (Chapter 5). A better understanding of the system dynamics may be found via detailed analysis of all the potential control loops. Factors of interest may be determined by an analysis of this type,

which could subsequently be used as controls in an experimental design. As with the sensitivity analysis suggestions above, a larger sample size would also be beneficial.

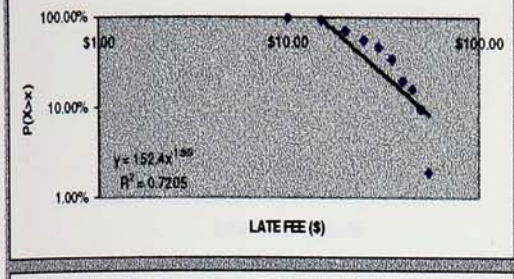
ALTERNATE AGENT ARCHITECTURES.

The Federative or Hybrid Agent Architecture developed in the CAS system may be improved via some of the above suggestions. Perhaps alternative Agent Architectures should be investigated as well. A study of Hierarchical vs. Heterarchical vs. Hybrid Agent Architectures would potentially reveal additional insights.

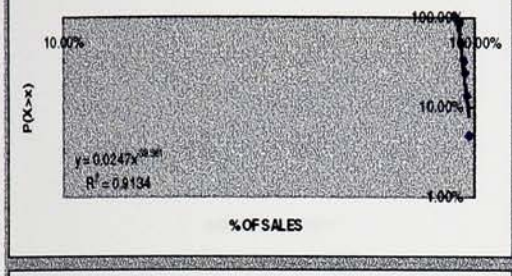
APPENDIX A - CAS FAST REPLICATIONS

CAS FAST REP 1

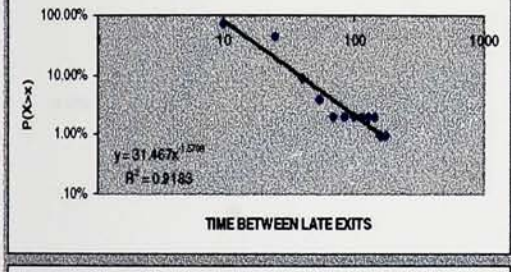
CAS.F.1 - LATE FEE PER LATE JOB



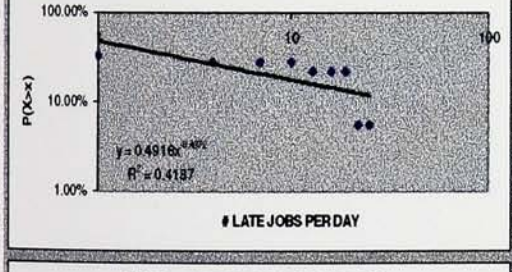
CAS.F.1 - % OF ACTUAL SALES FOR LATE JOBS



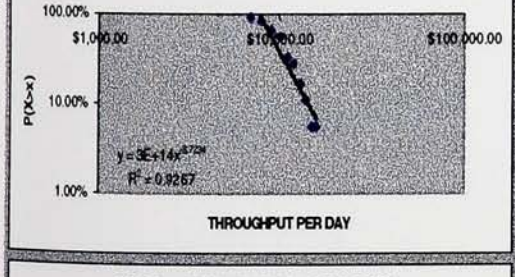
CAS.F.1 - TIME BETWEEN LATE EXITS



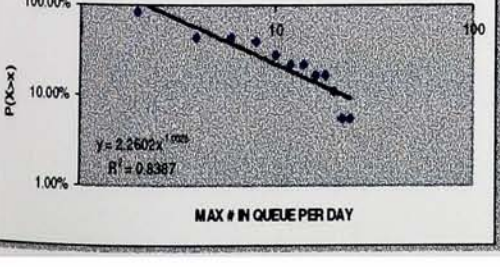
CAS.F.1 - # LATE JOBS PER DAY



CAS.F.1 - THROUGHPUT PER DAY

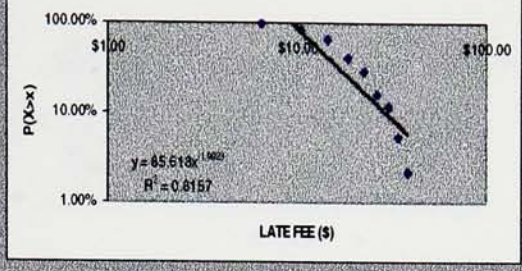


CAS.F.1 - MAX # IN QUEUE PER DAY



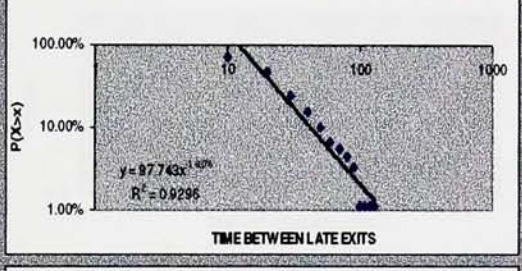
CAS FAST REP 2

CAS.F.2 - LATE FEE PER LATE JOB

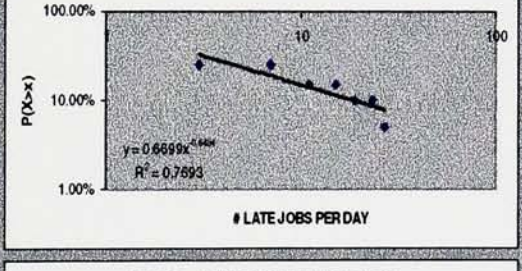


CAS.F.2 - % OF ACTUAL SALES FOR LATE JOBS

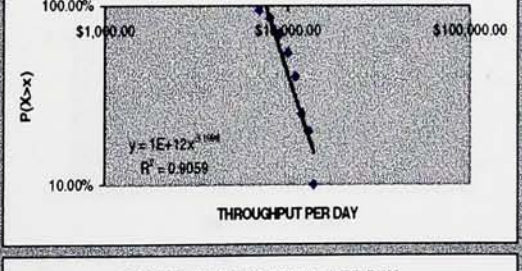
CAS.F.2 - TIME BETWEEN LATE EXITS



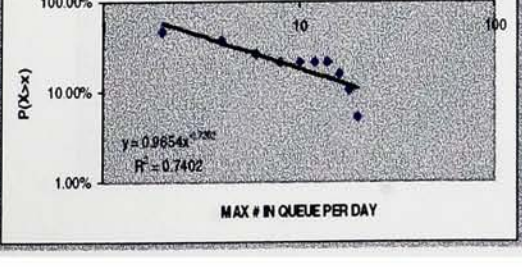
CAS.F.2 - # LATE JOBS PER DAY



CAS.F.2 - THROUGHPUT PER DAY



CAS.F.2 - MAX # IN QUEUE PER DAY



CAS FAST REP 3

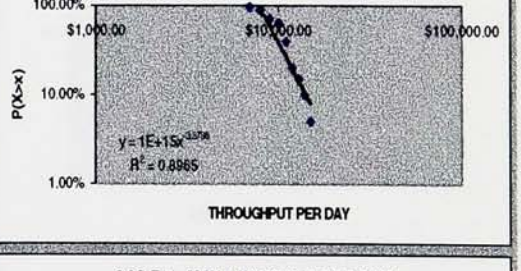
METRIC N.A.

METRIC N.A.

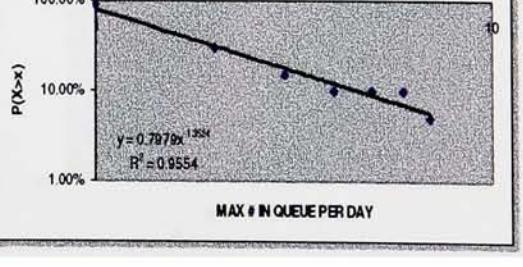
METRIC N.A.

METRIC N.A.

CAS.F.3 - THROUGHPUT PER DAY

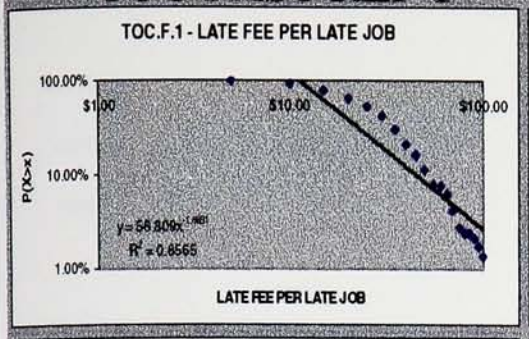


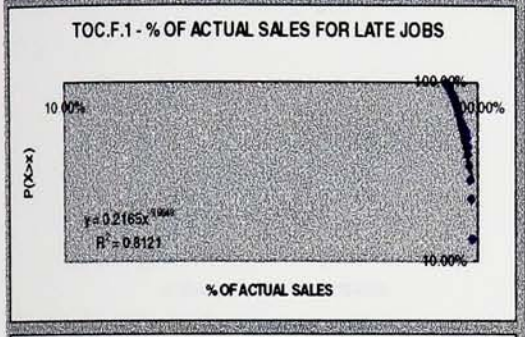
CAS.F.3 - MAX # IN QUEUE PER DAY

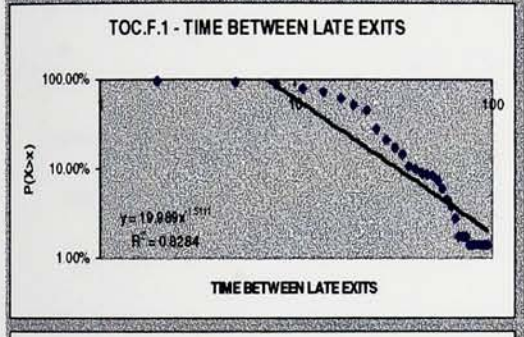


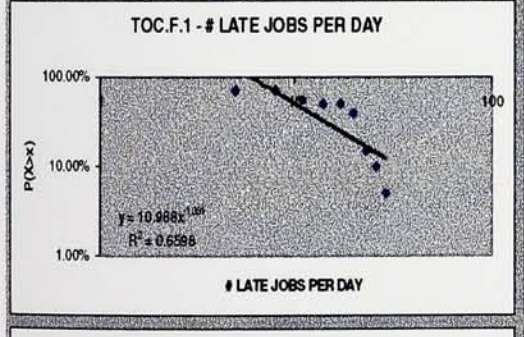
APPENDIX A - TOC FAST REPLICATIONS

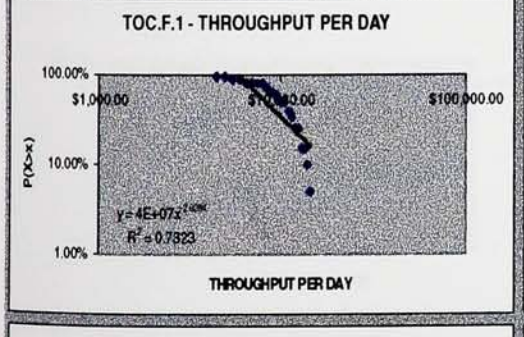
TOC FAST REP 1

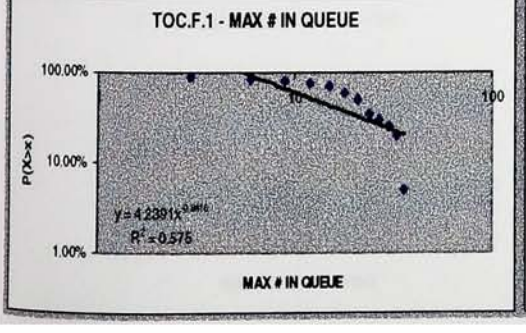




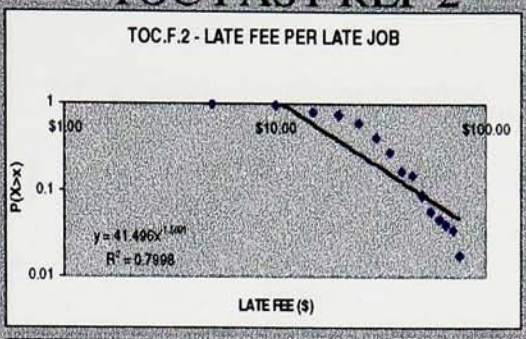


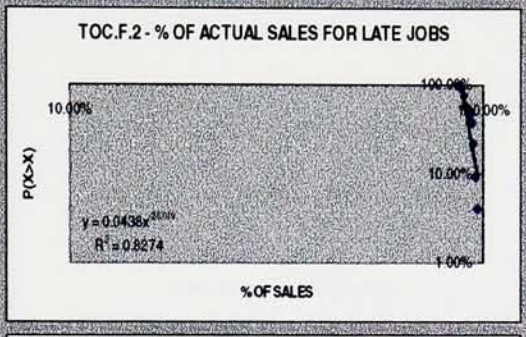


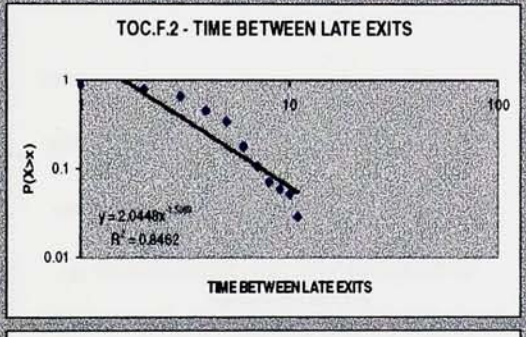


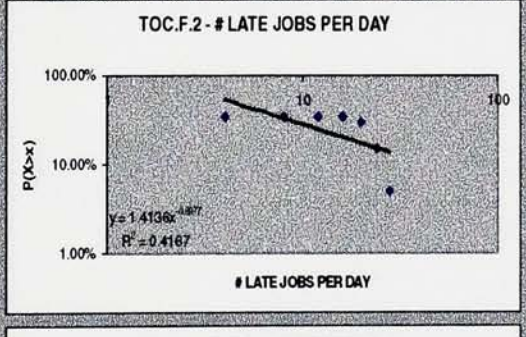


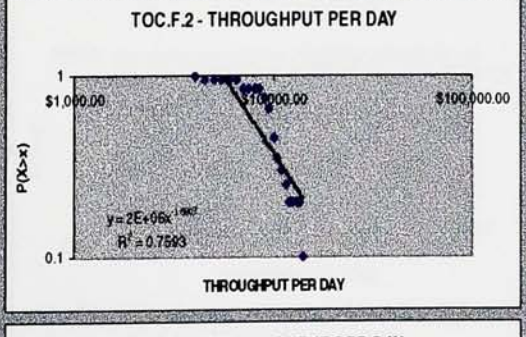
TOC FAST REP 2

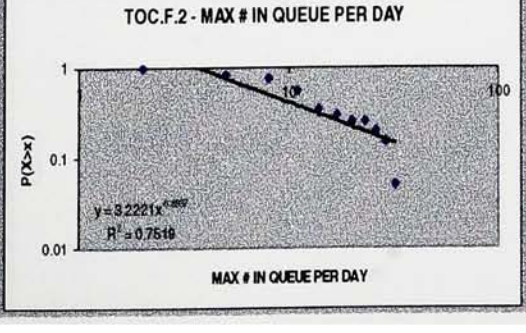




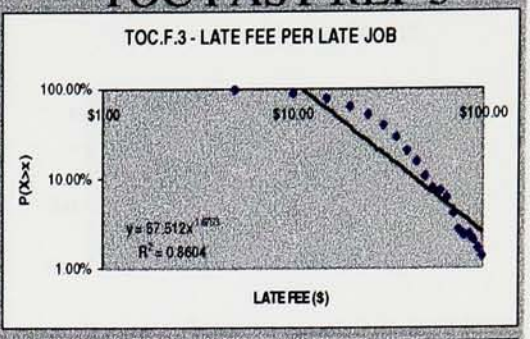


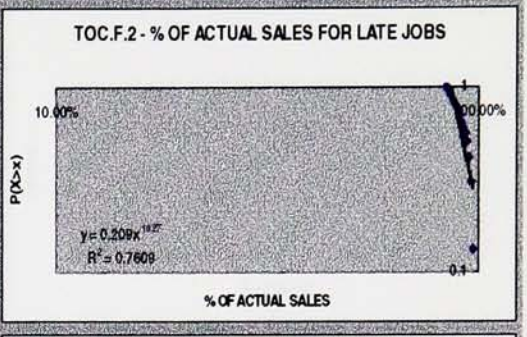


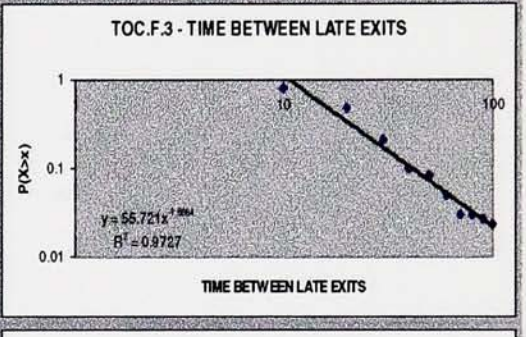


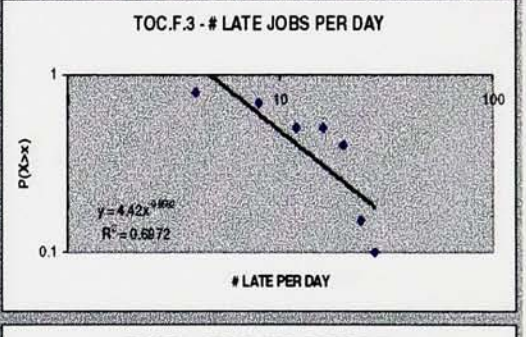


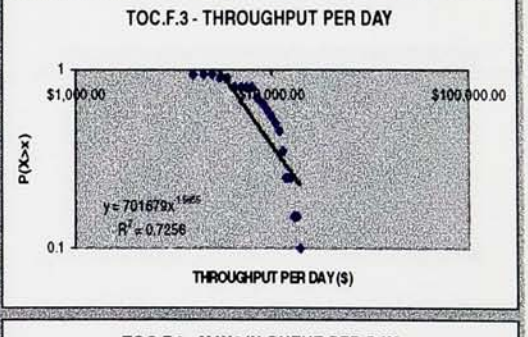
TOC FAST REP 3

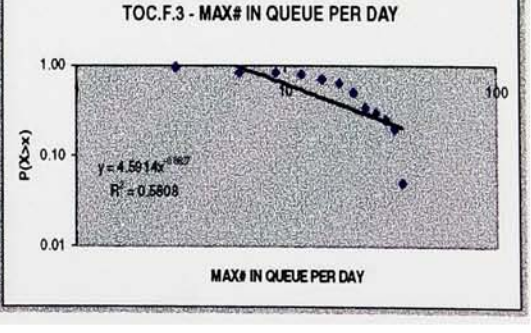






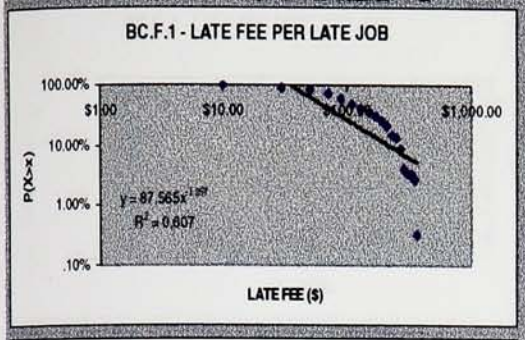


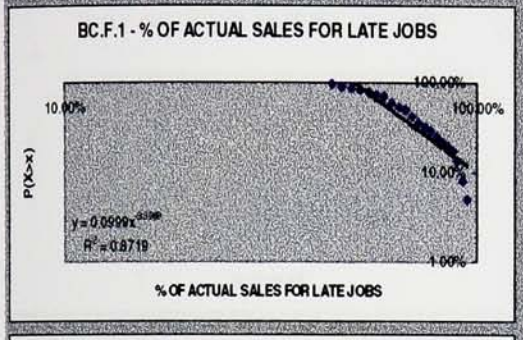


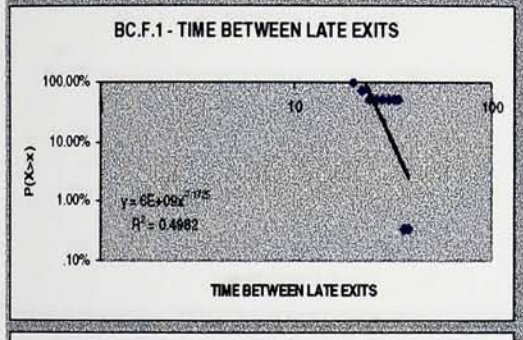


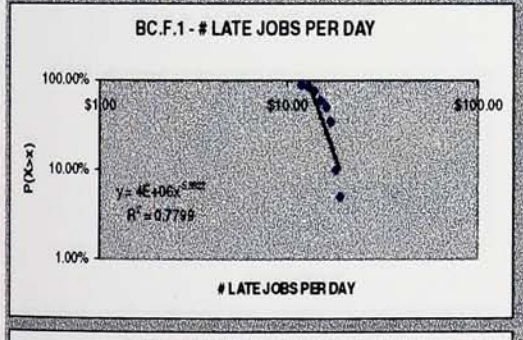
APPENDIX A - BC FAST REPLICATIONS

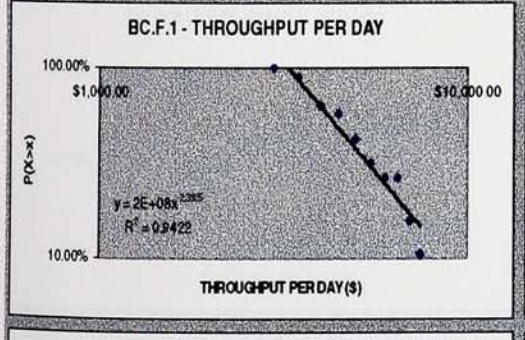
BC FAST REP 1

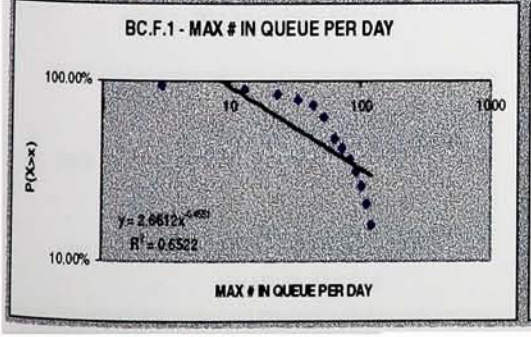




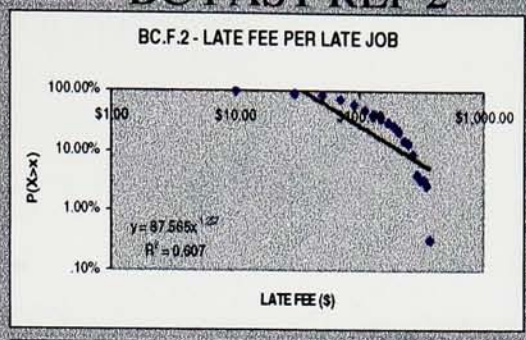


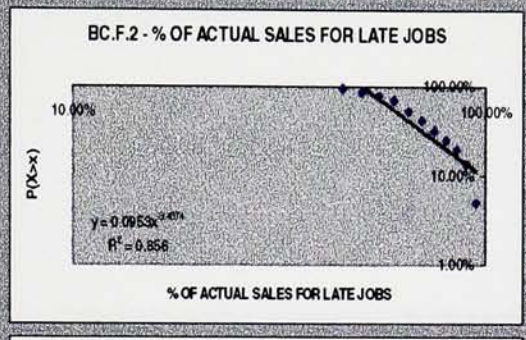


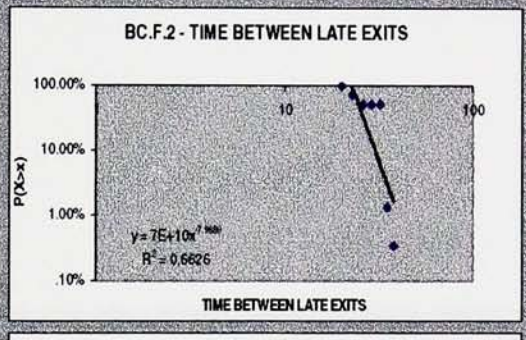


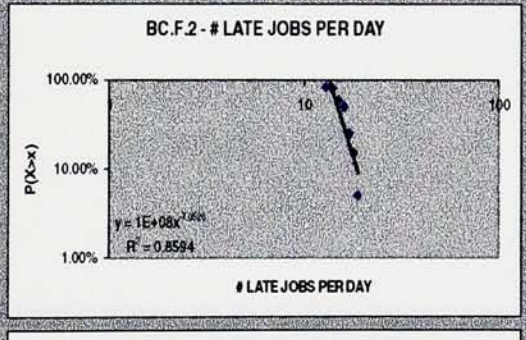


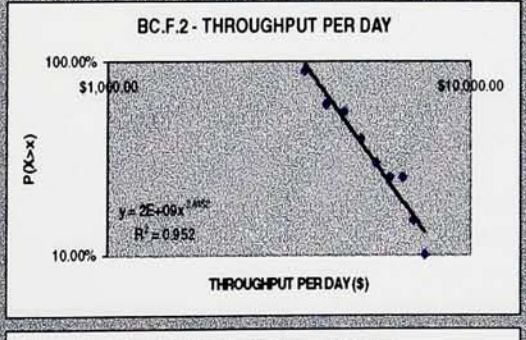
BC FAST REP 2

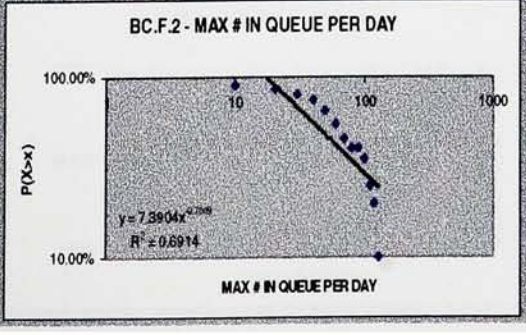




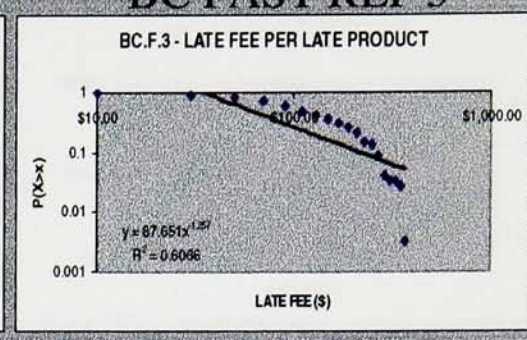


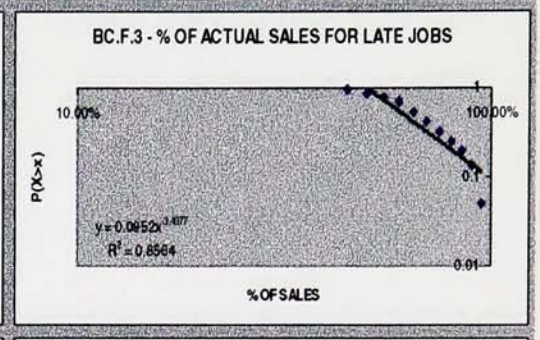


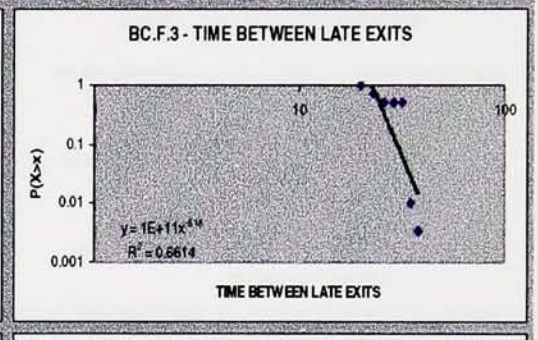


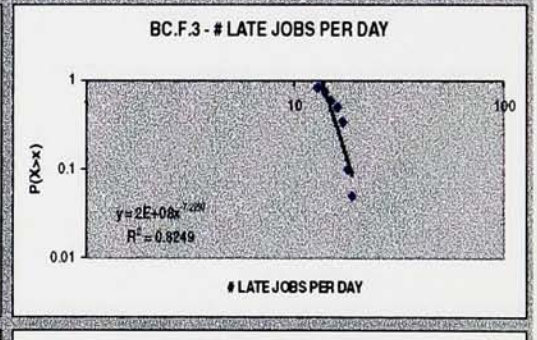


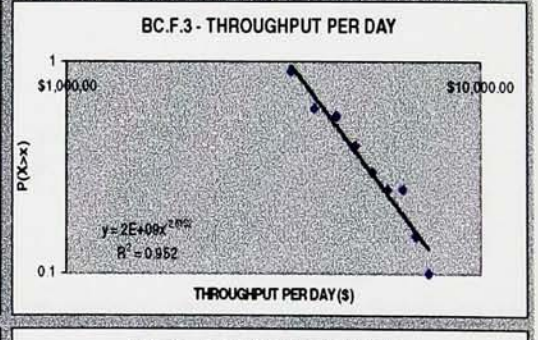
BC FAST REP 3

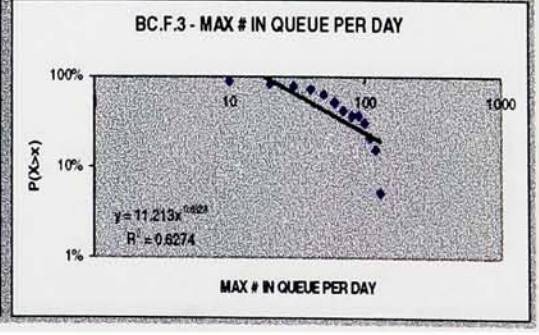






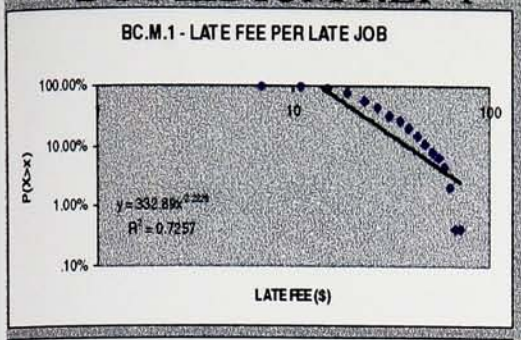


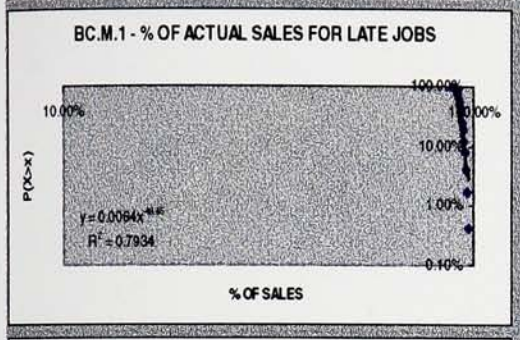


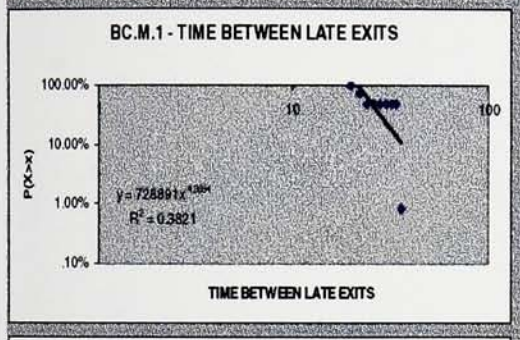


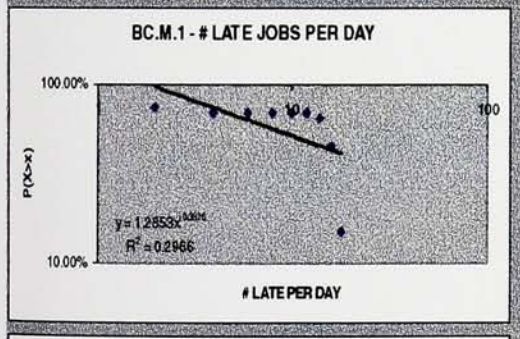
APPENDIX A - BC, TOC, CAS MEDIUM (ONE REPLICATION EACH)

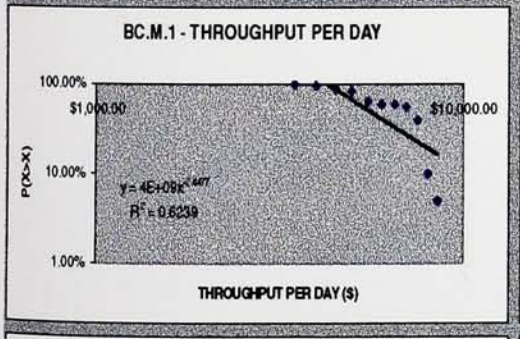
BC MEDIUM REP 1

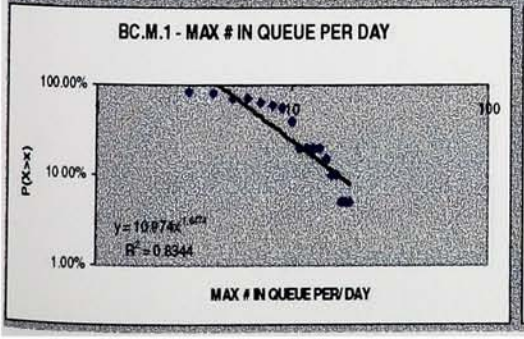












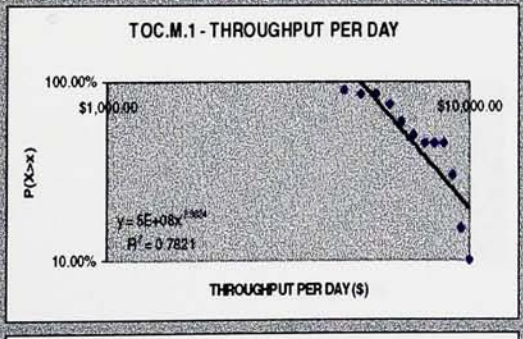
TOC MEDIUM REP 1

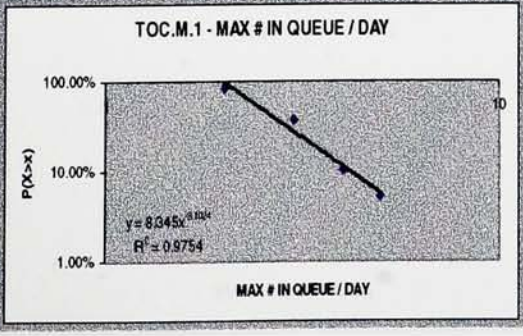
METRIC N.A.

METRIC N.A.

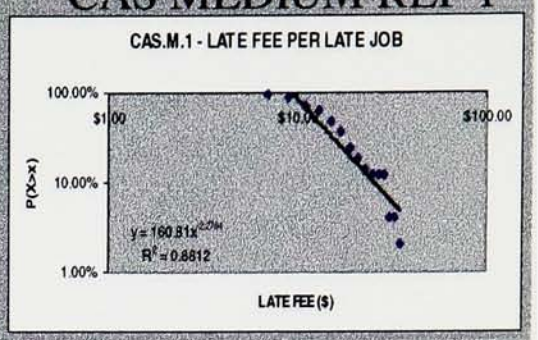
METRIC N.A.

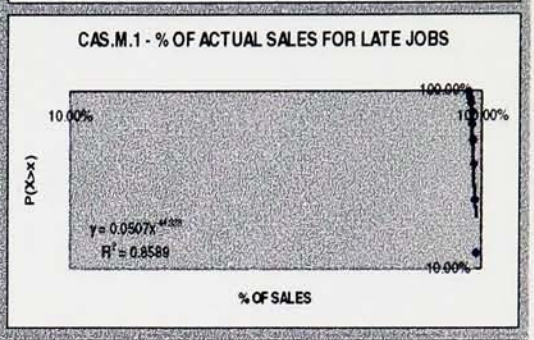
METRIC N.A.

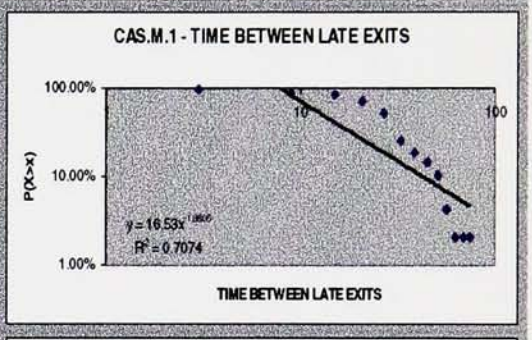


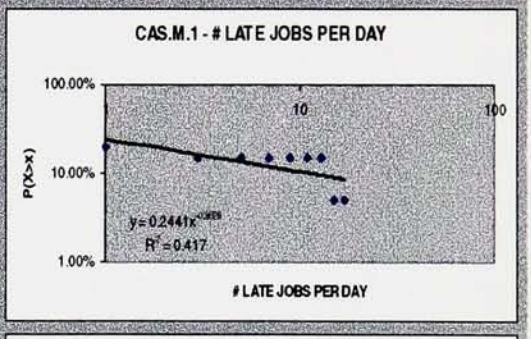


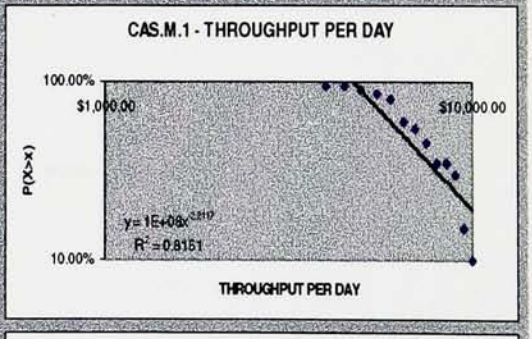
CAS MEDIUM REP 1

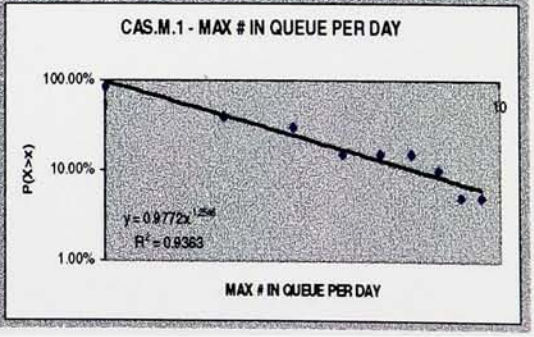










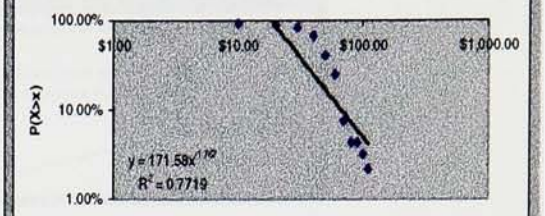
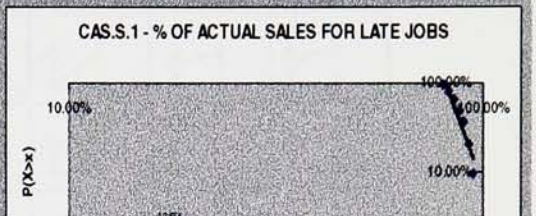
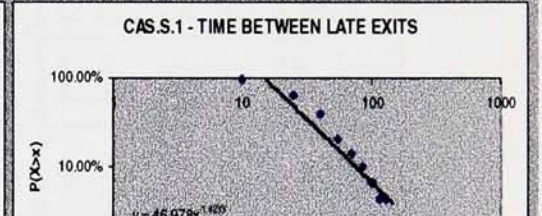
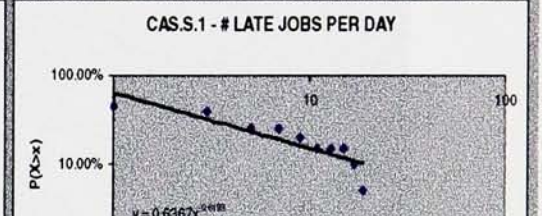
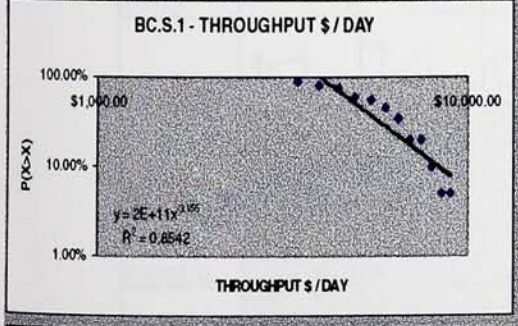
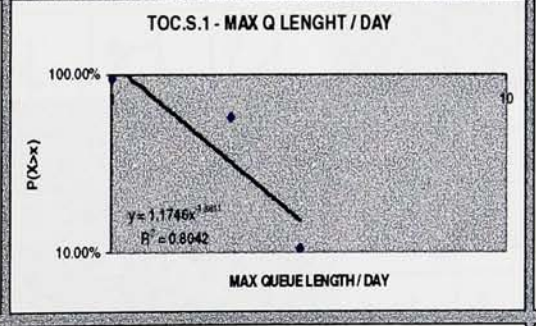
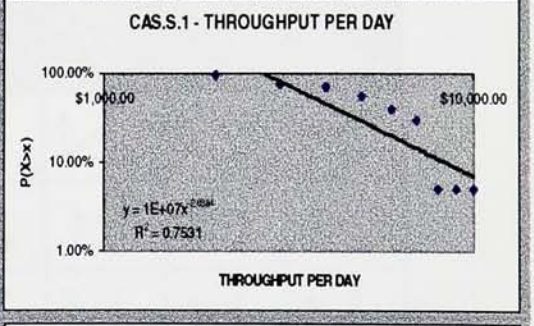
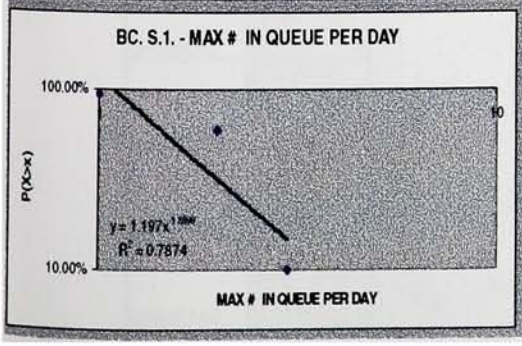
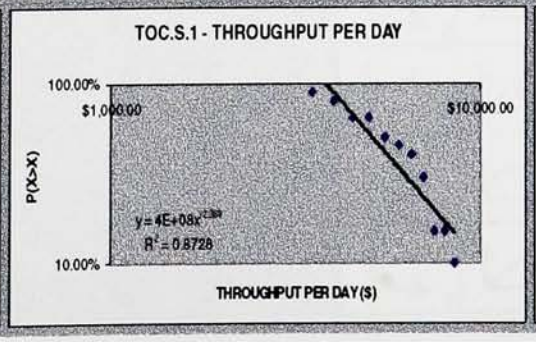
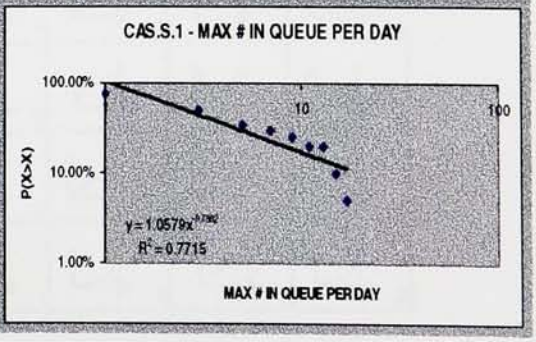


APPENDIX A - BC, TOC, CAS SLOW (ONE REPLICATION EACH)

BC SLOW REP 1

TOC SLOW REP 1

CAS SLOW REP 1

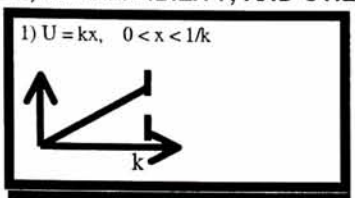
BC SLOW REP 1	TOC SLOW REP 1	CAS SLOW REP 1
METRIC N.A.	METRIC N.A.	
METRIC N.A.	METRIC N.A.	
METRIC N.A.	METRIC N.A.	
METRIC N.A.	METRIC N.A.	
		
		

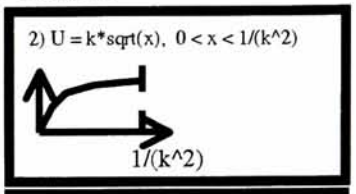
APPENDIX B - ASSEMBLER AGENT PAY RATES, WA ELIGIBILITY, AND UTILITY CURVES

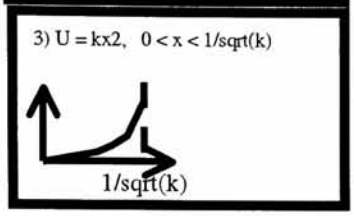
3 operator types	Hourly wage	10 stations	# of each operator
alpha	$ 25.00	1-10	6
beta	$ 18.00	1-5	6
gamma	$ 20.00	6-10	6

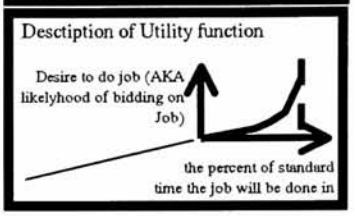
18 operators

a1	b1	g1
a2	b2	g2
a3	b3	g3
a4	b4	g4
a5	b5	g5
a6	b6	g6









	Utility functions per station									
Operator	WA1	WA2	WA3	WA4	WA5	WA6	WA7	WA8	WA9	WA10
ALPHA1	1	3	2	3	3	2	2	1	2	2
ALPHA2	2	2	1	2	1	3	3	2	2	2
ALPHA3	2	1	2	1	3	2	3	3	2	2
ALPHA4	2	3	2	1	3	2	3	1	2	1
ALPHA5	1	3	2	2	1	1	3	2	2	3
ALPHA6	2	3	2	1	3	2	3	2	2	2
BETA1	3	3	2	2	1	0	0	0	0	0
BETA2	2	1	2	1	3	0	0	0	0	0
BETA3	1	3	3	1	3	0	0	0	0	0
BETA4	3	2	2	1	3	0	0	0	0	0
BETA5	2	2	1	2	1	0	0	0	0	0
BETA6	2	1	3	2	3	0	0	0	0	0
GAMMA1	0	0	0	0	0	2	2	1	2	1
GAMMA2	0	0	0	0	0	3	3	2	2	1
GAMMA3	0	0	0	0	0	2	1	1	2	3
GAMMA4	0	0	0	0	0	2	1	2	1	3
GAMMA5	0	0	0	0	0	2	2	1	3	1
GAMMA6	0	0	0	0	0	2	1	3	1	3

Appendix C - Maximum Acceptable and Ideal Bid Per Product Per Location Justification

Process times at WA 1-10 for products 1-4

Process Time Array	WA1	WA2	WA3	WA4	WA5	WA6	WA7	WA8	WA9	WA10	Cum.Process Time	Sales $
Product 1	24	23	20	30	0	27	25	30	20	35	234	650
Product 2	12	0	12	18	14	17	0	18	12	21	124	350
Product 3	21	22	0	30	0	20	25	30	21	35	204	250
Product 4	14	15	14	20	16	19	17	21	14	24	174	500
Tot. P.T./ Station	71	60	46	98	30	83	67	99	67	115		437.5
Ave. P.T./Station	17.75	20	15.3333	24.5	15	20.75	22.3333	24.75	16.75	28.75		

Percentage of total processing time each WA contributes for each product type

10%	10%	9%	13%	0%	12%	11%	13%	9%	15%
10%	0%	10%	15%	11%	14%	0%	15%	10%	17%
10%	11%	0%	15%	0%	10%	12%	15%	10%	17%
8%	9%	8%	11%	9%	11%	10%	12%	8%	14%

(1) If only Alpha Agents produced this product the cost per product would be:
(2) If only Beta or Gamma Agents produced this product the cost per product would be:
(3) The percentage of the Sales $ spent on Labor for all Alphas of all Betas and Gammas:
(4) Maximum and ideal percentages of sales $ The system should spend (based on the output in (3))
(5) The Sales $ multiplied by the values in (4)
(6) Cost per location (these values would be used throughout the system if the processing times were more uniform from WA to WA)

	(1) hi (alpha only)	(2) lo (beta/ gamma only)	(3) percentage of total sales with all alpha (most expensive) all beta & gamma (least expensive)	(4) maximum& ideal levels/ prod	(5) acceptable system cost	(6) acceptable per location cost
product 1			15.00%	12%	$ 78.00	$ 7.80
	$ 97.50	$ 73.43	11.30%	9%	$ 58.75	$ 5.87
product 2			14.76%	12%	$ 41.33	$ 4.13
	$ 51.67	$ 39.07	11.16%	9%	$ 31.25	$ 3.13
product 3			34.00%	27%	$ 68.00	$ 6.80
	$ 85.00	$ 63.83	25.40%	20%	$ 50.91	$ 5.09
product 4			14.50%	12%	$ 58.00	$ 5.80
	$ 72.50	$ 54.83	10.97%	9%	$ 43.87	$ 4.39

The maximum acceptable price per product is equal to the Percentage of total processing time each WA contributes for each product type multiplied by the top value in (4)

Maximum acceptable price per product per location

Product 1	$ 8.00	$ 7.67	$ 6.67	$ 10.00	$ -	$ 9.00	$ 8.33	$ 10.00	$ 6.67	$ 11.67
Product 2	$ 4.00	$ -	$ 4.00	$ 6.00	$ 4.67	$ 5.67	$ -	$ 6.00	$ 4.00	$ 7.00
Product 3	$ 7.00	$ 7.33	$ -	$ 10.00	$ -	$ 8.67	$ 8.33	$ 10.00	$ 7.00	$ 11.67
Product 4	$ 4.67	$ 5.00	$ 4.67	$ 6.67	$ 5.33	$ 6.33	$ 5.67	$ 7.00	$ 4.67	$ 8.00

The ideal price per product is equal to the Percentage of total processing time each WA contributes for each product type multiplied by the bottom value in (4)

Ideal price per product per location

Product 1	$ 6.03	$ 5.77	$ 5.02	$ 7.53	$ -	$ 6.78	$ 6.28	$ 7.53	$ 5.02	$ 8.79
Product 2	$ 3.02	$ -	$ 3.02	$ 4.54	$ 3.53	$ 4.28	$ -	$ 4.54	$ 3.02	$ 5.29
Product 3	$ 5.24	$ 5.49	$ -	$ 7.49	$ -	$ 4.99	$ 6.24	$ 7.49	$ 5.24	$ 8.73
Product 4	$ 3.53	$ 3.78	$ 3.53	$ 5.04	$ 4.03	$ 4.79	$ 4.29	$ 5.29	$ 3.53	$ 6.05

Appendix D Base Case Library

Base Case Variable Library

ID	INTEGER/REAL	TYPE	DEFINITION
COUNT_V	Integer	Time Series, Time Weighted	#counter for incrementing the ID_AT attribute
DAILY_LATE_FEE_TOTAL_V	Real	Time Series, Observation Based	#for an 8 hour duration, this variable calculates the total system late fees
DAILY_NET_PROFIT_V	Real	Time Series, Observation Based	#how much money the system made each day
DAILY_THROUGHPUT_TOTAL_V	Real	Time Series, Observation Based	#how throughput the system generates each day
DAY_COUNT_V	Integer	Time Series, Time Weighted	keeps track of # of days system has run
MAX_Q_LENGTH_PER_DAY_V	Integer	Time Series, Observation Based	longest length of the incoming job queue per day
NUM_LATE_COUNTER_V	Integer	Time Series, Time Weighted	row identifier for output array that counts the number late
NUMBER_PRODUCTS_LATE_PER_DAY_V	Integer	Time Series, Observation Based	#how many late jobs there were today
NUMBER_PRODUCTS_LATE_V	Integer	Time Series, Observation Based	#Increments by 1for each product late through the system. At the end of the day the NUMBER_PRODUCTS_LATE_PER_DAY_V is set =number_products_late_v, then the number_products_late_v is zeroed out for the next day's calculation
PROD1_ACCUM_TIME_IN_SYS_V	Real	Time Series, Time Weighted	#captures the accumulated time in the system for the average time in system calculation
PROD1_AVERAGE_TIME_IN_SYS_V	Real	Time Series, Time Weighted	average time product one spends in the system
PROD1_COMPLETE_V	Integer	Time Series, Observation Based	#Counts the number of a certain product type that
PROD2_AVERAGE_TIME_IN_SYS_V	Real	Time Series, Time Weighted	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD2_COMPLETE_V	Integer	Time Series, Observation Based	#counts the number of a certain product type that are completed and leaves the system
PROD3_AVERAGE_TIME_IN_SYS_V	Real	Time Series, Time Weighted	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD3_COMPLETE_V	Integer	Time Series, Observation Based	#counts the number of a certain product type that is completed and leaves the system
PROD4_AVERAGE_TIME_IN_SYS_V	Real	Time Series, Time Weighted	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD4_COMPLETE_V	Integer	Time Series, Observation Based	#counts the number of a certain product type that is completed and leaves the system
Q_LENGTH_V	Integer	Time Series, Observation Based	used to keep track of current # in the incoming job queue,
Q1_ID_v	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q1_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
Q2_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q2_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
Q3_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q3_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
Q4_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q4_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
Q5_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q5_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
Q7_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
Q7_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
STAGE1_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
STAGE1_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
STAGE2_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
STAGE2_PROD_V	Integer	Time Series, Time Weighted	#displays which prod type is at a given location
SUB2_READY_V	Integer	Time Series, Time Weighted	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB3_READY_V	Integer	Time Series, Time Weighted	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB5_READY_V	Integer	Time Series, Time Weighted	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB7_READY_V	Integer	Time Series, Time Weighted	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUM_LATE_FEES_V	Real	Time Series, Observation Based	#accumulates the sum of late fees. Sum of late fees = sum of late fees + late_fee_at (if applicable)
SUM_THROUGHPUT_V	Real	Time Series, Observation Based	#Increments by each throughput throughout the system. At the end of the day the daily through is set =sum_throughput_v, then the sum of throughput variable is zeroed out for the next day's calculation
TIME_OF_LATE_EXIT_V	Real	Time Series, Observation Based	#self explanitory
TRUNCATED_AMOUNT_LATE_V	Integer	Time Series, Observation Based	#amount late v chopped down into an integer
WA1 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA1 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA10 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA10 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA2 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA2 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA3 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA3 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA4 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA4 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA4_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
WA5 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA5 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA6 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA6 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA6_ID_V	Integer	Time Series, Time Weighted	#displays which ID_AT is at a given location
WA7 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA7 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA8 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA8 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far
WA9 USED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX
WA9 UTILIZED V	Real	Time Series, Time Weighted	hours of work done at WORK AREAX/ hours system has run so far

Appendix D Base Case Library

Base Case Arrival Library

ENTITY	LOCATION	QTY EACH	FIRST TIME	OCCURRENCES	FREQUENCY	DESCRIPTION
DUMMY	HOME_LOCATION	1	0	INF	480 MIN	EVERY EIGHT HOUR STATISTICS COLLECTION FOR POWER LAW AND SYSTEM MEASURES

Base Case Array Library

ARRAY NAME	ROWS	COL.S	TYPE	DESCRIPTION
PROCESS_TIME_ARRAY	5	19	Real	Process times per product per work area
OUTPUT_ARRAY	480	22	Real	All System Metrics and Power Law data are output to this array

Base Case Macros Library

ID	Value	DESCRIPTION
LatePercentage_M	0.000208333	Prorated to the minute 10%/day late penalty
PROD1_PRICE_M	650	SALES PRICE FOR PROD1
PROD2_PRICE_M	350	SALES PRICE FOR PROD2
PROD3_PRICE_M	250	SALES PRICE FOR PROD3
PROD4_PRICE_M	500	SALES PRICE FOR PROD4
SLACKTIME_M	120	MINUTES AFTER LATE DATE LATE PENALTY BEGINS ACCRUING

Base Case Attribute Library

ID	TYPE	DEFINITION
PROD_AT	Integer	Id that discriminates between product types
ID_AT	Integer	Unique ID for every entity that enters the system
STARTQ_AT	Real	Captures time entity enters system/ IncJobQ
TIME_IN_SYS_AT	Real	Captures the time a product is in the entire sys. (Including Q)
TIME_IN_Q_AT	Real	Captures the time a product is in the IncJobQ
START_PROC_AT	Real	Captures the time a product begins processing
TIME_IN_PROC_AT	Real	Captures the time a product is in processing until completion
SALES_AT	Real	Captures the dollar value of sales - overhead - materials (fictitious amount)
LATE_FEE_AT	Real	Captures the late fee for an individual product based on how late it actually is
DUE_DATE_AT	Real	Read in Attribute from input file that lets the system know when the jobs are due
THROUGHPUT_AT	Real	Calculates individual throughput = sales_id - late_fee_at
PERCENTAGE_OF_TOTAL_SALES_AT	Real	ratio of (SALES_AT-LATE_FEE_AT)/SALES_AT
TIME_SINCE_LAST_LATE_FEE_AT	Real	amount of time passed since last late fee (no return means not late)

Base Case Resource Library

RESOURCE	QTY	COST ($)/MIN	DESCRIPTION
alpha1	1	.4166666667/min	Used at WA6
alpha2	1	.4166666667/min	Used at WA4
alpha3	1	.4166666667/min	Used at WA10
alpha4	1	.4166666667/min	Not used
beta1	1	.3/min	Used at WA1
beta2	1	.3/min	Used at WA2
beta3	1	.3/min	Used at WA3
beta4	1	.3/min	Used at WA5
gamma1	1	.33333333333/min	Used at WA7
gamma2	1	.33333333333/min	Used at WA8
gamma3	1	.33333333333/min	Used at WA9
gamma4	1	.33333333333/min	Not used

Appendix E Theory of Constraints Library

Theory of Constraints Variable Library

ID	TYPE	DESCRIPTION
AMOUNT_LATE_V	Real	# = DUE_DATE_AT - CLOCK (DAY)
COUNT_V	Integer	#counter for incrementing the ID_AT attribute
CURRENT_CONSTRAINT_V	Integer	IDENTIFIES WHICH WORK AREA IS THE CONSTRAINT
DAILY_LATE_FEE_TOTAL_V	Real	#for an 8 hour duration, this variable calculates the total system late fees
DAILY_NET_PROFIT_V	Real	#how much money the system made each day
DAILY_THROUGHPUT_TOTAL_V	Real	#how throughput the system generates each day
DAY_COUNT_V	Integer	keeps track of how many days the system has run
EXTRA_RESOURCES_WA1_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA10_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA2_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA3_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA4_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA5_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA6_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA7_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA8_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
EXTRA_RESOURCES_WA9_V	Integer	QUANTITY OF EXTRA RESOURCES AT A PARTICULAR STATION
JOB_RELEASE_V	Integer	Used in the toc model to emulate the drum beat of the drum buffer rope signaling of more material
MAX_Q_LENGTH_PER_DAY_V	Integer	Captures the maximum queue length for an eight hour period
NEXT_PAUSE_TIME_V	Integer	tells the system when the current day's data collection will be conducted
NUM_LATE_COUNTER_V	Integer	Row identifier for the output array. Puts Power law data for late products on a new line
NUMBER_PRODUCTS_LATE_PER_DAY_V	Integer	#how many late jobs there were today
NUMBER_PRODUCTS_LATE_V	Integer	#Increments by 1for each product late through the system. At the end of the day the NUMBER_PRODUCTS_LATE_PER_DAY_V is set =number_products_late_v, then the number_products_late_v is zeroed out for the next day's calculation
OVERFLOW_WA1_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA10_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA2_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA3_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA4_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA5_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA6_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA7_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA8_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
OVERFLOW_WA9_V	Integer	Amount of processing left for a given job at the end of the data collection period (day)
PROD1_ACCUM_TIME_IN_SYS_V	Real	#captures the accumulated time in the system for the average time in system calculation
PROD1_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD1_COMPLETE_V	Integer	#Counts the number of a certain product type that
PROD2_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD2_COMPLETE_V	Integer	#counts the number of a certain product type that are completed and leaves the system
PROD3_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD3_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
PROD4_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD4_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
Q_LENGTH_V	Integer	counts the length of the queue, used in the max q length per day calculation for system to system comparison
Q1_ID_v	Integer	#displays which ID_AT is at a given location
Q1_PROD_V	Integer	#displays which ID_AT is at a given location
Q2_ID_V	Integer	#displays which ID_AT is at a given location
Q2_PROD_V	Integer	#displays which ID_AT is at a given location
Q3_ID_V	Integer	#displays which ID_AT is at a given location
Q3_PROD_V	Integer	#displays which ID_AT is at a given location
Q4_ID_V	Integer	#displays which ID_AT is at a given location
Q5_ID_V	Integer	#displays which ID_AT is at a given location
Q5_PROD_V	Integer	#displays which ID_AT is at a given location
Q7_ID_V	Integer	#displays which ID_AT is at a given location
Q7_PROD_V	Integer	#displays which ID_AT is at a given location
STAGE1_ID_V	Integer	#displays which ID_AT is at a given location
STAGE1_PROD_V	Integer	#displays which ID_AT is at a given location
STAGE2_ID_V	Integer	#displays which ID_AT is at a given location
STAGE2_PROD_V	Integer	#displays which ID_AT is at a given location
SUB2_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB3_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB5_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB7_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUM_LATE_FEES_V	Real	#accumulates the sum of late fees. Sum of late fees = sum of late fees + late_fee_at (if applicable)
SUM_THROUGHPUT_V	Real	#Increments by each throughput throughout the system. At the end of the day the daily through is set =sum_throughput_v, then the sum of throughput variable is zeroed out for the next day's calculation
TIME_OF_LATE_EXIT_V	Real	#self explanatory
TRUNCATED_AMOUNT_LATE_V	Integer	#amount late v chopped down into an integer
WA1_USED_V	Real	amount of time the work area is being used
WA1_UTILIZED_V	Real	WORK AREA UTILIZATION
WA10_USED_V	Real	amount of time the work area is being used
WA10_UTILIZED_V	Real	WORK AREA UTILIZATION
WA2_USED_V	Real	amount of time the work area is being used
WA2_UTILIZED_V	Real	WORK AREA UTILIZATION
WA3_USED_V	Real	amount of time the work area is being used
WA3_UTILIZED_V	Real	WORK AREA UTILIZATION
WA4_ID_V	Integer	#displays which ID_AT is at a given location
WA4_USED_V	Real	amount of time the work area is being used
WA4_UTILIZED_V	Real	WORK AREA UTILIZATION
WA5_USED_V	Real	amount of time the work area is being used
WA5_UTILIZED_V	Real	WORK AREA UTILIZATION
WA6_ID_V	Integer	shows which ID_AT is at a work area
WA6_ID_V	Integer	#displays which ID_AT is at a given location
WA6_USED_V	Real	amount of time the work area is being used
WA6_UTILIZED_V	Real	WORK AREA UTILIZATION
WA7_USED_V	Real	amount of time the work area is being used
WA7_UTILIZED_V	Real	WORK AREA UTILIZATION
WA8_USED_V	Real	amount of time the work area is being used
WA8_UTILIZED_V	Real	WORK AREA UTILIZATION
WA9_USED_V	Real	amount of time the work area is being used
WA9_UTILIZED_V	Real	WORK AREA UTILIZATION

Appendix E Theory of Constraints Library

Theory of Constraints Arrivals Library

ENTITY	LOCATION	QTY EACH	FIRST TIME	OCCURRENCES	FREQUENCY	DESCRIPTION
DUMMY	HOME_LOCATION	1	0	INF	480 MIN	Check every eight hours for Power Law and System Measures

Theory of Constraints Array Library

ID	ROWS	COLS	TYPE	DESCRIPTION
PROCESS_TIME_ARRAY	5	19	Real	Product process times per Work area
OUTPUT_ARRAY	480	22	Real	All System Measure and Power Law data output to this array

Theory of Constraints Macro Library

ID	VALUE	DESCRIPTION
LatePercentage_M	0.000208333	Prorated to the minute 10%/day late penalty
PROD1_PRICE_M	650	SALES PRICE FOR PROD1
PROD2_PRICE_M	350	SALES PRICE FOR PROD2
PROD3_PRICE_M	250	SALES PRICE FOR PROD3
PROD4_PRICE_M	500	SALES PRICE FOR PROD4
ALPHA_PAY_M	25	HOURLY LABOR COST
BETA_PAY_M	18	HOURLY LABOR COST
GAMMA_PAY_M	20	HOURLY LABOR COST
ZETA_PAY_M	25	HOURLY LABOR COST
Time_M	480	number of minutes in a day
AVE_PT_M	5	THE ROW IN THE ARRAY WITH AVERAGE PROCESS TIME PER STATION
BUFFER_TIME_M	2	Value used to accurately calculate the utilization of resources
SLACKTIME_M	120	MINUTES AFTER LATE DATE LATE PENALTY BEGINS ACCRUING

Theory of Constraints Attribute Library

ID	TYPE	DEFINITION
PROD_AT	Integer	Id that discriminates between product types
ID_AT	Integer	Unique ID for every entity that enters the system
STARTQ_AT	Real	Captures time entity enters system/ IncJobQ
TIME_IN_SYS_AT	Real	Captures the time a product is in the entire sys. (Including Q)
TIME_IN_Q_AT	Real	Captures the time a product is in the IncJobQ
START_PROC_AT	Real	Captures the time a product begins processing
TIME_IN_PROC_AT	Real	Captures the time a product is in processing until completion
SALES_AT	Real	Captures the dollar value of sales - overhead - materials (fictitious amount)
LATE_FEE_AT	Real	Captures the late fee for an individual product based on how late it actually is
DUE_DATE_AT	Real	Read in Attribute from input file that lets the system know when the jobs are due
THROUGHPUT_AT	Real	Calculates individual throughput = sales_id - late_fee_at
PERCENTAGE_OF_TOTAL_SALES_AT	Real	ratio of (SALES_AT-LATE_FEE_AT)/SALES_AT
TIME_SINCE_LAST_LATE_FEE_AT	Real	amount of time passed since last late fee (no return means not late)

Theory of Constraints Resource Library

RESOURCE	QTY	COST ($)/M	DESCRIPTION
alpha1	1	.416666667/	Used at WA6
alpha2	1	.416666667/	Used at WA4
alpha3	1	.416666667/	Used at WA10
alpha4	1	.416666667/	Not used
beta1	1	.3/MIN	Used at WA1
beta2	1	.3/MIN	Used at WA2
beta3	1	.3/MIN	Used at WA3
beta4	1	.3/MIN	Used at WA5
gamma1	1	.333333333	Used at WA7
gamma2	1	.333333333	Used at WA8
gamma3	1	.333333333	Used at WA9
gamma4	1	.333333333	Not used
ZETA1	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA2	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA3	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA4	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA5	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA6	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA7	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA8	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA9	1	.416666667/	Can work at WA1-WA10 as added capacity
ZETA10	1	.416666667/	Can work at WA1-WA10 as added capacity
INTERVENTION AGENT	1	.5/MIN	Modeler representing a scheduler in the system

Appendix F Complex Adaptive System Library

Complex Adaptive System Variable Library

ID	INTEGER/REAL	DEFINITION
ACCUMULATED_ASSIGNMENTS_OF_ACCEPTABLE_OPERATORS	Integer	counts how many times an "acceptble" operator was assigned to a job
ACCUMULATED_ASSIGNMENTS_OF_BUSY_OPERATORS_V	Integer	counts how many times an "busy" operator was assigned to a job
ACCUMULATED_ASSIGNMENTS_OF_IDEAL_OPERATORS_V	Integer	counts how many times a "ideal" operator was assigned to a job
DAY_COUNT_V	Integer	keeps track of how many days the system has run
Q4_ID_V	Integer	#displays which ID_AT is at a given location
Q4_prod_v	Integer	#displays which id_at is at a given location
NUM_LATE_COUNTER_V	Integer	counter that keeps track of late prodcuts for array purposes Each unique number late becomes a row in the output array
AMOUNT_LATE_V	Real	# = DUE_DATE_AT - CLOCK (DAY)
SUM_LATE_FEES_V	Real	#accumulates the sum of late fees Sum of late fees = sum of late fees + late_fee_at (if applicable)
TRUNCATED_AMOUNT_LATE_V	Integer	#amount late v chopped down into an integer
PROD1_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD2_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD3_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD4_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD1_ACCUM_TIME_IN_SYS_V	Real	#captures the accumulated time in the system for the average time in system calculation
COUNT_V	Integer	#counter for incrementing the ID_AT attribute
PROD1_COMPLETE_V	Integer	#Counts the number of a certain product type that
PROD2_COMPLETE_V	Integer	#counts the number of a certain product type that are completed and leaves the system
PROD3_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
PROD4_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
Q1_ID_v	Integer	#displays which ID_AT is at a given location
Q1_PROD_V	Integer	#displays which id_at is at a given location
Q2_ID_V	Integer	#displays which ID_AT is at a given location
Q2_PROD_V	Integer	#displays which id_at is at a given location
Q3_ID_V	Integer	#displays which ID_AT is at a given location
Q3_PROD_V	Integer	#displays which id_at is at a given location
Q5_ID_V	Integer	#displays which ID_AT is at a given location
Q5_PROD_V	Integer	#displays which id_at is at a given location
Q7_ID_V	Integer	#displays which ID_AT is at a given location
Q7_PROD_V	Integer	#displays which id_at is at a given location
STAGE1_ID_V	Integer	#displays which id_at is at a given location
STAGE1_PROD_V	Integer	#displays which id_at is at a given location
STAGE2_ID_V	Integer	#displays which id_at is at a given location
STAGE2_PROD_V	Integer	#displays which id_at is at a given location
WA4_ID_V	Integer	#displays which ID_AT is at a given location
WA6_ID_V	Integer	#displays which ID_AT is at a given location
DAILY_LATE_FEE_TOTAL_V	Real	#for an 8 hour duration, this variable calculates the total system late fees
NUMBER_PRODUCTS_LATE_PER_DAY_V	Integer	#how many late jobs there were today
DAILY_NET_PROFIT_V	Real	#how much money the system made each day
DAILY_THROUGHPUT_TOTAL_V	Real	#how throughput the system generates each day
NUMBER_PRODUCTS_LATE_V	Integer	#Increments by 1for each product late through the system. At the end of the day the NUMBER_PRODUCTS_LATE_PER_DAY_V is set =number_products_late_v, then the number_products_late_v is zeroed out for the next day's calculation
SUM_THROUGHPUT_V	Real	#Increments by each throughput throughout the system. At the end of the day the daily through is set =sum_throughput_v, then the sum of throughput variable is zeroed out for the next day's calculation
SUB2_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB3_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB5_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB7_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
TIME_OF_LATE_EXIT_V	Real	#self explanitory
REARRANGE_Q_COST_V	Real	2 hour cost to rearrange queue
MAX_Q_LENGTH_PER_DAY_V	Integer	biggest queue length per day
BID_RATE_V	Real	either 1 0 or 0.8 depending on how much the que arrangement cost in the past 2 hours
BID_FLAG_V	Integer	global flag variable that allows only one entity at one location bid at one time
WA10 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA2 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA3 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA4 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA5 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA6 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA7 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA8 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA9 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA1 USED V	Real	hours of work done at WORK AREAX
WA10 USED V	Real	hours of work done at WORK AREAX
WA2 USED V	Real	hours of work done at WORK AREAX
WA3 USED V	Real	hours of work done at WORK AREAX
WA4 USED V	Real	hours of work done at WORK AREAX
WA5 USED V	Real	hours of work done at WORK AREAX
WA6 USED V	Real	hours of work done at WORK AREAX
WA7 USED V	Real	hours of work done at WORK AREAX
WA8 USED V	Real	hours of work done at WORK AREAX
WA9 USED V	Real	hours of work done at WORK AREAX
LAST_ADDED_V	Integer	keeps track of the last resource added, so the system knows which Assembler Agent to add or remove next
NQ_V	Integer	number in the queue
ID_AT_EXITS_V	Integer	Number of exits out of the system
NUM_RES_WA1_V	Integer	number of resources at a WA
NUM_RES_WA10_V	Integer	number of resources at a WA
NUM_RES_WA2_V	Integer	number of resources at a WA
NUM_RES_WA3_V	Integer	number of resources at a WA
NUM_RES_WA4_V	Integer	number of resources at a WA
NUM_RES_WA5_V	Integer	number of resources at a WA
NUM_RES_WA6_V	Integer	number of resources at a WA
NUM_RES_WA7_V	Integer	number of resources at a WA
NUM_RES_WA8_V	Integer	number of resources at a WA
NUM_RES_WA9_V	Integer	number of resources at a WA
SUM_Q_REARRANGE_COST_V	Real	total cost to rearrange queue
NUMBER_IN_SYSTEM_V	Integer	TOTAL NUMBER IN QUEUE AND IN PROCESSING
BUMP_DOWN_SUM_V	Integer	total number of jobs shuffeled
NUMBER_ASSIGNED_BY_SUPERVISOR_V	Integer	keeps track of how many times the supervisor has to assign assembler agents. It is used to decide if more resources or less resources are needed for the next day (processing in WA10)

Appendix F Complex Adaptive System Library

Complex Adaptive System Variable Library

ID	INTEGER/REAL	DEFINITION
ACCUMULATED_ASSIGNMENTS_OF_ACCEPTABLE_OPERATORS	Integer	counts how many times an "acceptble" operator was assigned to a job
ACCUMULATED_ASSIGNMENTS_OF_BUSY_OPERATORS_V	Integer	counts how many times an "busy" operator was assigned to a job
ACCUMULATED_ASSIGNMENTS_OF_IDEAL_OPERATORS_V	Integer	counts how many times a "ideal" operator was assigned to a job
DAY_COUNT_V	Integer	keeps track of how many days the system has run
Q4_ID_V	Integer	#displays which ID_AT is at a given location
Q4_prod_v	Integer	#displays which id_at is at a given location
NUM_LATE_COUNTER_V	Integer	counter that keeps track of late prodcuts for array purposes Each unique number late becomes a row in the output array
AMOUNT_LATE_V	Real	# = DUE_DATE_AT - CLOCK (DAY)
SUM_LATE_FEES_V	Real	#accumulates the sum of late fees. Sum of late fees = sum of late fees + late_fee_at (if applicable)
TRUNCATED_AMOUNT_LATE_V	Integer	#amount late v chopped down into an integer
PROD1_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD2_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD3_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD4_AVERAGE_TIME_IN_SYS_V	Real	#calculates average time in system by dividing the accumulated time in system by the number of products of a certain type that have left the system
PROD1_ACCUM_TIME_IN_SYS_V	Real	#captures the accumulated time in the system for the average time in system calculation
COUNT_V	Integer	#counter for incrementing the ID_AT attribute
PROD1_COMPLETE_V	Integer	#Counts the number of a certain product type that
PROD2_COMPLETE_V	Integer	#counts the number of a certain product type that are completed and leaves the system
PROD3_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
PROD4_COMPLETE_V	Integer	#counts the number of a certain product type that is completed and leaves the system
Q1_ID_v	Integer	#displays which ID_AT is at a given location
Q1_PROD_V	Integer	#displays which id_at is at a given location
Q2_ID_V	Integer	#displays which ID_AT is at a given location
Q2_PROD_V	Integer	#displays which id_at is at a given location
Q3_ID_V	Integer	#displays which ID_AT is at a given location
Q3_PROD_V	Integer	#displays which id_at is at a given location
Q5_ID_V	Integer	#displays which ID_AT is at a given location
Q5_PROD_V	Integer	#displays which id_at is at a given location
Q7_ID_V	Integer	#displays which ID_AT is at a given location
Q7_PROD_V	Integer	#displays which id_at is at a given location
STAGE1_ID_V	Integer	#displays which id_at is at a given location
STAGE1_PROD_V	Integer	#displays which id_at is at a given location
STAGE2_ID_V	Integer	#displays which id_at is at a given location
STAGE2_PROD_V	Integer	#displays which id_at is at a given location
WA4_ID_V	Integer	#displays which ID_AT is at a given location
WA6_ID_V	Integer	#displays which ID_AT is at a given location
DAILY_LATE_FEE_TOTAL_V	Real	#for an 8 hour duration, this variable calculates the total system late fees
NUMBER_PRODUCTS_LATE_PER_DAY_V	Integer	#how many late jobs there were today
DAILY_NET_PROFIT_V	Real	#how much money the system made each day
DAILY_THROUGHPUT_TOTAL_V	Real	#how throughput the system generates each day
NUMBER_PRODUCTS_LATE_V	Integer	#Increments by 1for each product late through the system. At the end of the day the NUMBER_PRODUCTS_LATE_PER_DAY_V is set =number_products_late_v, then the number_products_late_v is zeroed out for the next day's calculation
SUM_THROUGHPUT_V	Real	#Increments by each throughput throughout the system. At the end of the day the daily through is set =sum_throughput_v, then the sum of throughput variable is zeroed out for the next day's calculation
SUB2_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB3_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB5_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
SUB7_READY_V	Integer	#indicates whether or not a location is ready to send an entity ahead to be matched and joined
TIME_OF_LATE_EXIT_V	Real	#self explanitory
REARRANGE_Q_COST_V	Real	2 hour cost to rearrange queue
MAX_Q_LENGTH_PER_DAY_V	Integer	biggest queue length per day
BID_RATE_V	Real	either 1 0 or 0.8 depending on how much the que arrangement cost in the past 2 hours
BID_FLAG_V	Integer	global flag variable that allows only one entity at one location bid at one time
WA10 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA2 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA3 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA4 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA5 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA6 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA7 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA8 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA9 UTILIZED V	Real	hours of work done at WORK AREAX/ hours system has run so far
WA1 USED V	Real	hours of work done at WORK AREAX
WA10 USED V	Real	hours of work done at WORK AREAX
WA2 USED V	Real	hours of work done at WORK AREAX
WA3 USED V	Real	hours of work done at WORK AREAX
WA4 USED V	Real	hours of work done at WORK AREAX
WA5 USED V	Real	hours of work done at WORK AREAX
WA6 USED V	Real	hours of work done at WORK AREAX
WA7 USED V	Real	hours of work done at WORK AREAX
WA8 USED V	Real	hours of work done at WORK AREAX
WA9 USED V	Real	hours of work done at WORK AREAX
LAST_ADDED_V	Integer	keeps track of the last resource added, so the system knows which Assembler Agent to add or remove next
NQ_V	Integer	number in the queue
ID_AT_EXITS_V	Integer	Number of exits out of the system
NUM_RES_WA1_V	Integer	number of resources at a WA
NUM_RES_WA10_V	Integer	number of resources at a WA
NUM_RES_WA2_V	Integer	number of resources at a WA
NUM_RES_WA3_V	Integer	number of resources at a WA
NUM_RES_WA4_V	Integer	number of resources at a WA
NUM_RES_WA5_V	Integer	number of resources at a WA
NUM_RES_WA6_V	Integer	number of resources at a WA
NUM_RES_WA7_V	Integer	number of resources at a WA
NUM_RES_WA8_V	Integer	number of resources at a WA
NUM_RES_WA9_V	Integer	number of resources at a WA
SUM_Q_REARRANGE_COST_V	Real	total cost to rearrange queue
NUMBER_IN_SYSTEM_V	Integer	TOTAL NUMBER IN QUEUE AND IN PROCESSING
BUMP_DOWN_SUM_V	Integer	total number of jobs shuffeled
NUMBER_ASSIGNED_BY_SUPERVISOR_V	Integer	keeps track of how many times the supervisor has to assign assembler agents. It is used to decide if more resources or less resources are needed for the next day (processing in WA10)