

DEVELOPMENT OF A METHODOLOGY FOR HYBRID
METAMODELING OF HIERARCHICAL
MANUFACTURING SYSTEMS
WITHIN A SIMULATION
FRAMEWORK

By

DAVID BRUCE PRATT

Bachelor of Science
Oklahoma State University
Stillwater, Oklahoma
1976

Master of Engineering
Oklahoma State University
Stillwater, Oklahoma
1977

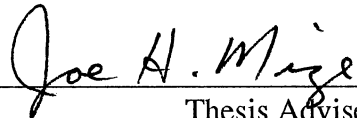
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1992

1992 D

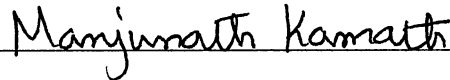
Thesis
1992 D
P913d

DEVELOPMENT OF A METHODOLOGY FOR HYBRID
METAMODELING OF HIERARCHICAL
MANUFACTURING SYSTEMS
WITHIN A SIMULATION
FRAMEWORK

Thesis Approved:



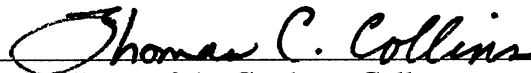
Thesis Adviser











Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my appreciation to all those who have helped me realize a long held dream, completing this dissertation and earning my Ph.D. First and foremost, this work is dedicated to my wife Jan and two children Brian and Kristi. When I thank God for the blessings in my life, you three top the list. The sacrifices you have made without complaint over the last three years is more than I had a right to ask for. The encouragement and support you have offered is more than I could have hoped for. To each of you I extend my love and heartfelt thanks for never letting me lose sight of the truly important things in life and for helping me prove that sometimes to make dreams come true, you have to be courageous enough to listen to your heart.

I would also like to thank my parents, Harold and Lahoma Pratt. Your ever present support and encouragement has been a source of strength. I believe that everything of lasting worth is built upon a solid foundation. Thank you for instilling in me a foundation based on love and a set of values that has stood the test of time.

Special recognition also goes to my doctoral committee. To my advisor Dr. Joe Mize and to committee member Dr. Ken Case, words cannot express the debt I owe the two of you. When I examine the list of people who have had a significant impact on molding me into what I am today, your names are prominent. In the years to come, I will judge my career a success if I can look back and know that I have provided students with the same inspiration, guidance, and example that you have provided me. To committee members Dr. Palmer Terrell, Dr. Manjunath Kamath, and Dr. Charles Bacon, thank you for your guidance, counsel, and encouragement. I have learned many things from each of you and for that I thank you.

Finally, to my friends, the residents of the CIM Center, I am glad our paths crossed. I will always remember and cherish our friendships. In particular I would like to thank Phil Farrington and Chuda Basnet. We "non-traditional students" have to stick together. Thanks for blazing the trail in front of me.

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION.....	1
	Motivation For The Research.....	1
	Motivation - A Broader Perspective.....	3
	Overview Of The Dissertation.....	5
II.	STATEMENT OF THE PROBLEM.....	7
	Introduction.....	7
	Simulation In A Manufacturing Context.....	9
	Fundamental Issues Of Simulation Modeling.....	12
	Object Oriented Modeling.....	13
	Top-Down Versus Bottom-Up Modeling.....	15
	The Impact Of OOM On The Fundamental Issues.....	17
	Addressing The Abstraction Issue: Metamodeling.....	18
	Hybrid Modeling.....	22
	Unanswered Questions.....	23
III.	BACKGROUND OF THE STUDY.....	24
	Introduction.....	24
	Metamodeling Of Hierarchical Systems.....	24
	Hybrid Modeling.....	28
	Performance Modeling Of Manufacturing Systems.....	30
	Summary.....	45
IV.	STATEMENT OF RESEARCH.....	48
	Research Goal.....	48
	Research Objectives.....	48
	Research Assumptions.....	52
	Research Contributions.....	53
V.	RESEARCH METHODOLOGY.....	55
	Performance Measure.....	55
	Research Plan.....	55
	Observation-Based Metamodel Scenarios.....	60
	Queueing Network Metamodel Scenarios.....	63

Chapter	Page
Plant Level Prototype Validation.....	64
Observation-Based Metamodel Development Procedure.....	66
Queueing Network Metamodel Development Procedure.....	68
Metamodel Initialization Procedure.....	69
VI. OBJECT ORIENTED REPRESENTATION.....	71
Introduction.....	71
Object Oriented Classes.....	71
Changes Made To The Environment.....	74
Conducting An OOM Experiment.....	76
VII. METAMODEL SELECTION PROCEDURE.....	81
Introduction.....	81
Assessment Of Candidate Workcenters.....	82
Availability Of A Metamodel.....	85
VIII. EVALUATION OF THE METHODOLOGY.....	87
Introduction.....	87
Experimental Evaluation.....	87
Characteristics Common To All Scenarios.....	88
Scenario QN1 - Tandem Queueing Network.....	89
Scenario QN2 - Tree Queueing Network.....	101
Scenario OB1 - State Dependent Routings.....	107
Scenario OB2 - Multiple Concurrent Resources.....	119
Scenario OB3 - Machine Breakdowns.....	127
Scenario OB4 - Finite Queue Capacity.....	135
Inter-Scenario Comparisons.....	144
Computer Execution Times.....	146
Summary of Experimental Results.....	148
IX. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS.....	150
Introduction.....	150
Research Summary.....	150
Research Contributions.....	155
Recommendations For Future Research.....	156
BIBLIOGRAPHY.....	160
APPENDIXES.....	169
APPENDIX A - SMALLTALK-80 CODE FOR ENVIRONMENT MODIFICATIONS.....	170

Chapter	Page
APPENDIX B - SIMULATION RUN DESIGN CONSIDERATIONS.....	204
APPENDIX C - ANALYSIS OF VARIANCE OF EXPERIMENTAL DATA.....	210
APPENDIX D - PERSPECTIVES ON COMPARING TWO DERIVED NUMBERS.....	230
APPENDIX E - EMPIRICAL GROUPED CUMULATIVE DISTRIBUTION FUNCTIONS.....	233
APPENDIX F - METAMODEL FILE SPECIFICATION.....	236
APPENDIX G - SAS PROGRAM FOR WORKCENTER LEVEL VALIDATION.....	242

LIST OF TABLES

Table	Page
I. Balance Equations For Figure 5.....	37
II. Balance Equations For Figure 6.....	39
III. Tandem Network Fast Simulation Recursion Equations.....	43
IV. Metamodeling Literature Overview.....	46
V. Hybrid Modeling Literature Overview.....	46
VI. Significant OOM Environment Changes.....	75
VII. OOM Routing Definition.....	79
VIII. Manufacturing Measures Of Performance.....	82
IX. Metamodeling Candidate Assessment.....	84
X. Scenario Identification.....	88
XI. ANOVA Summary For QN1 Plant Level Validation #2.....	98
XII. ANOVA Summary For QN1 Plant Level Validation #3.....	100
XIII. ANOVA Summary For QN1 Plant Level Validation #3a.....	101
XIV. ANOVA Summary For QN2 Plant Level Validation #2.....	105
XV. ANOVA Summary For QN2 Plant Level Validation #3.....	106
XVI. ANOVA Summary For QN2 Plant Level Validation #3a.....	106
XVII. ANOVA Summary For OB1 Plant Level Validation #2.....	118
XVIII. ANOVA Summary For OB1 Plant Level Validation #3.....	118
XIX. ANOVA Summary For OB1 Plant Level Validation #3a.....	119
XX. ANOVA Summary For OB2 Plant Level Validation #2.....	125
XXI. ANOVA Summary For OB2 Plant Level Validation #3.....	126

Table	Page
XXII. ANOVA Summary For OB2 Plant Level Validation #3a.....	127
XXIII. ANOVA Summary For OB3 Plant Level Validation #2.....	133
XXIV. ANOVA Summary For OB3 Plant Level Validation #3.....	134
XXV. ANOVA Summary For OB3 Plant Level Validation #3a.....	135
XXVI. Queue Capacities For OB4 Machines.....	136
XXVII. ANOVA Summary For OB4 Plant Level Validation #2.....	143
XXVIII. ANOVA Summary For OB4 Plant Level Validation #3.....	143
XXIX. ANOVA Summary For OB4 Plant Level Validation #3a.....	145
XXX. Summary Of Computer Execution Time Savings.....	147

LIST OF FIGURES

Figure	Page
1. Hierarchical Structure Of A Manufacturing System.....	12
2. Base Model Of A Hierarchical Manufacturing System.....	20
3. Metamodel Version Of A Hierarchical Manufacturing System.....	20
4. Metamodels And Experimental Frames.....	21
5. A Simple Open Jackson Queueing Network.....	37
6. A Simple Closed Jackson Queueing Network.....	38
7. Decision Based Validation.....	51
8. Observation-Based Metamodel Workcenter Structure.....	62
9. Basic Configuration Of Plant Model.....	64
10. Detailed Configuration Of Plant Model.....	65
11. Plot Of Observed Time-In-System Distribution.....	67
12. CDF Curves For A Workcenter Group.....	68
13. OOM Class Hierarchy Diagram For Metamodeling.....	73
14. OOM Plant Structure Definition.....	77
15. Scenario OB3 Plant Definition File.....	78
16. BOM Definition File For All Scenarios.....	78
17. Scenario OB3 Routing Definition File.....	79
18. Workcenter QN1 Structure.....	90
19. Plant Level Validation #1.....	94
20. Plant QN1 Validation #1 - Visual Inspection.....	95
21. Workcenter QN2 Structure.....	102

Figure	Page
22. Plant QN2 Validation #1 - Visual Inspection.....	104
23. Workcenter OB1 Structure.....	108
24. Plant OB1 Workcenter Validation - Rho 0.50.....	114
25. Plant OB1 Workcenter Validation - Rho 0.75.....	115
26. Plant OB1 Validation #1 - Visual Inspection.....	117
27. Plant OB2 Workcenter Validation - Rho 0.50.....	121
28. Plant OB2 Workcenter Validation - Rho 0.75.....	122
29. Plant OB2 Validation #1 - Visual Inspection.....	124
30. Plant OB3 Workcenter Validation - Rho 0.50.....	129
31. Plant OB3 Workcenter Validation - Rho 0.75.....	130
32. Plant OB3 Validation #1 - Visual Inspection.....	132
33. Plant OB4 Workcenter Validation - Rho 0.50.....	138
34. Plant OB4 Workcenter Validation - Rho 0.75.....	139
35. Plant OB4 Validation #1 - Visual Inspection.....	142

CHAPTER I

INTRODUCTION

Motivation For The Research

One primary motivation for this research is the author's desire to contribute to the advancement of simulation modeling of manufacturing systems from a practitioner's point of view. Simulation has proven to be an excellent vehicle for studying complex systems. It is perhaps the only viable tool for analyzing the detailed dynamic behavior of such systems. However, simulation is not without its disadvantages. Foremost on the list of disadvantages is that simulation can be costly, requiring large expenditures of time and resources for model construction, validation and execution.

Recent progress in the development of an advanced simulation environment at Oklahoma State University's (OSU) Center for Computer Integrated Manufacturing has resulted in several key advancements in both conceptual and methodological capabilities. Among these advancements are (1) the conceptualization and demonstration (in prototype form) of decision modularity through separate modeling constructs for physical, information, and control elements of a system and (2) the creation of a framework for highly reusable modeling enabled by a library of modeling primitives that can be retrieved and reused as desired. This approach to system modeling represents a change that is perhaps more revolutionary than evolutionary. It represents a major paradigm shift in the construction, utilization, and maintenance of models.

A significant element of this paradigm shift is the inherent adoption of a bottom-up rather than top-down view of model construction. This bottom-up process is a direct

result of a previously derived library of modeling primitives. Typically, the library of primitives includes generic representations for such things as machines and material handlers as well as company-specific constructs for particular lathes, robots, conveyors, AGVs, etc. By selecting modeling primitives from the library, interconnecting their input/output ports using routings, and implementing their decision processes through control policies, models of specific systems can be created.

If a particular combination of interconnected primitives (e.g., a representation of a workcenter) is useful in more than one situation, it too can be stored in the modeling database for subsequent use. This representation is no longer called a modeling primitive. It becomes a coupled model.

One significant advantage of having this library of reusable modeling primitives and coupled models is a reduction in the time required to build and validate complex models. This time advantage accrues because developing a model is no longer a "code from scratch" process. It is a "select and connect" process using previously validated building blocks. The obvious tradeoff is that considerable up-front work is required to create and validate the library. Additional on-going effort is required to maintain the validity of the library as the real system changes over time. This effort is most easily justified by viewing the modeling library as a new type of corporate asset; an asset that must remain congruent with the actual physical, information, and control assets of the enterprise.

The modeling approach described above raises an interesting question. What if the level of detail represented by the primitives and coupled models exceeds the level required for a particular experimental investigation? From one perspective, a modeler might be tempted to say "Who cares? Use it anyway. If the model provides more detailed information than required, no one is hurt." If computational resources are plentiful and time is not a constraint, then perhaps this perspective is acceptable. Unhappily, based on this practitioner's experience, neither of these is usually the case.

From a different perspective, a modeler might logically ask "Can some portion of the model be represented in a less detailed way to reduce model complexity while retaining essential behaviors?" This question is one of creating simplified yet valid "models of models" and using them effectively and efficiently in concert with a detailed simulation model. Expressed in the language of this research, it is a question of creating valid metamodels and using them in a hybrid modeling environment.

Addressing this question is a challenging task. While queueing network models offer insight into some manufacturing configurations, little controlled research has been done in an attempt to answer this question relative to complex manufacturing systems that are beyond the realm of queueing models. As a result, this area is a fruitful area for academic investigation. It is in search of insights in this area that the following research is offered.

Motivation - A Broader Perspective

The primary motivation for pursuing research in the area of hybrid metamodels is one of resource and time savings. In today's computing environment (hardware and software) this is a reasonable and justifiable issue. Execution times for complex simulation models are frequently measured in minutes or hours rather than seconds. When alternative scenarios are evaluated or search based optimization is pursued, total elapsed times may stretch across multiple hours or even days. Under such conditions, simulation modeling becomes restricted by resource and time constraints. Further, using simulation for on-line real-time decision making, an area where simulation has seen little use but has tremendous potential, is impractical due to lengthy execution time. These circumstances represent situations within which faster, less resource intensive simulation would have significant value.

While everything in the above paragraph is true, it represents a shortsighted view of computer-based simulation. Technologically, few areas are advancing faster than

computing hardware and software. Ultimately, faster microchips and superior processing paradigms (e.g., parallel processing) coupled with advances in simulation languages and methodology will reduce execution times by orders of magnitude. Uses of simulation that are today impractical will become realizable. Will model execution time then become a non-issue or will practitioners simply stretch the complexity boundary such that there will always be models that require "too much time and too many resources?" This is not an issue that can be answered here; only time will tell. However, just raising it casts doubt on using execution time as the sole justification for hybrid metamodels.

From a broader viewpoint two significant perspectives emerge. First, from a scholarly perspective, understanding and insight into the behavior of complex systems are gained during the creative process required for the development of metamodels. Second, from an information perspective, metamodels may be the only modeling option available if lack of information prohibits the creation of an appropriate coupled model from the modeling primitives in the library.

The scholarly perspective is perhaps best summarized by Ignall and Kolesar [1979, 232]. In summarizing the use of simulation to build new OR/MS theory they state:

. . . . new systems knowledge will not happen if papers reporting results of simulation studies continue to declare in effect "this system is too complicated for analytic models." What we have tried to stress is the potential value of asking "Is it really too complicated?" and following up on that question. That is, compare the simulation results with those of a related analytic model. If those simulation results do show the inadequacy of the analytical model, spend some time and effort hypothesizing a functional form for the model that would be adequate. Then test it by running new simulations. Iterate this process until you are satisfied.

Perhaps on reflection, this perspective outweighs the potential reduction in model execution time in terms of the most significant contribution that this research can make.

The second perspective, the information perspective, is based on the fact that despite best efforts, there are times when bottom-up modeling may not be appropriate. The bottom-up perspective inherently assumes that the database of modeling primitives contains either generic or company-specific representations of all components of the real system to be modeled. But, what if a representation is needed of a component that is so new that information regarding its performance is only known (or estimated) in gross terms? In this case a metamodel may be the best (or only) choice available for including the component in a system model. Subsequent availability of detailed information may permit the replacement of the metamodel with a detailed counterpart. This process exemplifies the more traditional top-down approach to modeling since the metamodel was developed prior to the detailed model. While the thrust of this research is on the development of metamodels from their detailed counterparts (the bottom-up approach), insights gained into interchanging metamodels and detailed models as well as insights into metamodel implementation should be useful in the top-down approach as well.

Overview Of The Dissertation

The remainder of this dissertation is presented in eight chapters plus a bibliography and appendices. Chapter II develops the problem statement in detail, within which, many of the points made above are explored more fully. Chapter III reviews the literature relevant to this research effort. This includes literature about performance modeling of manufacturing systems, metamodeling of hierarchical systems, and hybrid modeling. Chapter IV presents the statement of research by defining the research goal and research objectives. The chapter also presents the simplifying assumptions made in defining the research and the intended contributions of the effort. Chapter V discusses the research methodology including performance measures, experimental scenarios, and planned

metamodel development procedures. Chapter VI provides a brief overview of the object-oriented modeling environment within which this research is conducted. Chapters VII and VIII present the results of the investigation. Specifically, Chapter VII focuses on the metamodel selection procedure and Chapter VIII focuses on the results of the simulation experiments. Chapter IX is the summary and conclusions chapter that synthesizes the results of this effort and suggests directions that appear fruitful for additional investigation. The seven appendices provide supporting material including listings of computer code and rationalization of choices in several areas.

CHAPTER II

STATEMENT OF THE PROBLEM

Introduction

A system is a collection of interdependent elements which work cooperatively for the purpose of achieving a common goal. Frequently, a system is characterized by random, but statistically predictable, behavior. A model is a representation of a system. If the model is expressed mathematically as a set of logical and functional relationships, it is referred to as an abstract model. A computer-based simulation model is an abstract model implemented on a computer upon which experiments are conducted for the purpose of generating information useful in making decisions. Simulation modeling is only one alternative within a set of techniques which are useful in investigating the characteristics and behaviors of a system. At one end of this spectrum of techniques is analytical modeling, while at the other end is experimentation on the real system.

Analytical models employ techniques from stochastic processes and queueing theory to study system performance. They are generally the most efficient method of investigation if they are applicable. They frequently yield explicit information about the functional form of the relationships among system variables and, under some circumstances, can point to further analysis which will produce an optimal solution.

Unfortunately, some real systems of interest are so complex that formulating and solving an exact analytical model is either extremely difficult or impossible. Schriber [1987, 6], addressing the subject from perhaps an unduly negative perspective, states:

... the system being modeled must often be unduly distorted to fit a model amenable to analytic solution, and one can wind up with "the right solution for the wrong problem." (The right solution for the wrong problem is the wrong solution.)

Suri [1983] demonstrates a vastly more favorable outlook through his work showing that queueing network models are robust in many practical situations. In recent years, considerable interest has been shown in approximate solutions to analytical models. Although approximate, many situations that commonly occur in manufacturing systems can be modeled quite well [Bitran and Tirupati 1988; Kamath, Suri, and Sanders 1988; Segal and Whitt 1989].

At the other end of the spectrum of techniques, experimentation on the real system suffers from many practical limitations. First and foremost, the real system may not even exist. The system being investigated may be purely hypothetical. Other limiting factors are that it may not be economically feasible to suspend on-going operations in order to perform the experiments, it may be too dangerous and/or destructive to test the real system, and experimentation may take too long to complete to be of value.

Simulation modeling overcomes many of the disadvantages inherent in analytical modeling and experimentation on the real system. It permits realistic models of arbitrary complexity; it can represent either real or proposed systems; it effectively compresses time; it allows repetition of the same "history" so that alternatives can be equitably compared; and results are typically easier to "sell" to nontechnical managers than those of analytical solutions. Unfortunately, as stated in Chapter I, simulation is not without its disadvantages. Foremost on the list being the high expenditure of time and money.

As an analysis technique, simulation is recognized as a technique of high practical value. Numerous surveys have shown that it is one of the more popular and powerful tools available to help solve real problems [Shannon, Long, and Buckles 1980; Ledbetter and Cox 1977; Cook and Russell 1976; Paul 1991]. Of particular interest to this research

is the breadth and depth of manufacturing issues which have been studied using simulation [Schriber 1987; Law 1986].

In recent years, an approach which combines simulation with analytical results has emerged. This approach has come to be known as hybrid modeling [Shanthikumar and Sargent 1983]. Although hybrid models are typically not as accurate as pure simulation models, they take advantage of the speed of analytical methods and are thus more cost effective.

Even more recently, Chen and Chen [1990] and Kamath and Bhuskute [1991] have demonstrated the validity of fast simulation. Fast simulation involves replacing the traditional simulation event calendar with a set of recursive equations which can be solved very rapidly. For certain classes of queueing networks, fast simulation can save up to 80% of simulation run time. Research is currently being conducted at OSU's Center for Computer Integrated Manufacturing to explore a hybrid approach employing both fast and traditional simulation simultaneously within a simulation run.

The research to be conducted in this study most closely aligns with the hybrid approaches presented in the previous two paragraphs. While it does not exactly fit under the umbrella of either of the above approaches, it is pursued in the same spirit. That is, an attempt is made to take advantage of the speed offered by a derived analytical relationship within the framework of a manufacturing simulation model.

Simulation In A Manufacturing Context

A manufacturing enterprise is an excellent example of a system. It is a collection of interdependent elements (physical components, information components, and control policies). It exhibits random (usually statistically predictable) behavior. And, its elements work together to achieve a common goal (to manufacture products of acceptable

quality at a specified rate). Manufacturing systems are frequently too large and complex to be modeled in detail using analytical techniques.¹ Operational practicalities generally prohibit experimentation on the real system. Simulation is usually the analysis tool of choice.

Law [1986] points out that simulation has been used to address a wide variety of manufacturing issues. Among these issues are:

- (1) The need for and the quantity of equipment and personnel
 - (a) Number and type of machines
 - (b) Number, type and physical arrangement of material handlers and support equipment
 - (c) Location and size of inventory buffers
 - (d) Evaluation of change in product mix (impact of new products)
 - (e) Evaluation of the effect of a new piece of equipment on an existing manufacturing line
 - (f) Evaluation of capital investments
 - (g) Manpower requirements planning
- (2) Performance evaluation
 - (a) Throughput analysis
 - (b) Makespan analysis
 - (c) Bottleneck analysis
- (3) Evaluation of operational procedures
 - (a) Production scheduling (i.e., evaluating proposed policies for loading and sequencing machines)
 - (b) Evaluation of policies for component part or raw material inventory levels

¹As noted earlier, many queuing theorists and practitioners may disagree with this statement (see for instance Koenigsberg [1991]). Their disagreement is respectfully acknowledged.

- (c) Evaluation of control strategies (e.g., for an AGV system or an FMS)
- (d) Reliability analysis (e.g., effect of planned maintenance)
- (e) Evaluation of quality control policies

In addressing these issues, Law [1986] goes on to state that the measures of performance most often used in manufacturing simulation studies include:

- (1) Throughput (number of jobs produced per unit of time)
- (2) Time in system for all jobs (makespan)
- (3) Time jobs spend in queue(s)
- (4) Time that jobs spend being transported
- (5) Sizes of WIP inventories
- (6) Utilization of equipment and personnel
- (7) Proportion of time that a machine is broken, blocked, or starved
- (8) Proportion of jobs produced which must be reworked or scrapped
- (9) Return on investment of a new or modified manufacturing system
- (10) Payback period of a new or modified manufacturing system

Simulation models of manufacturing systems are frequently formulated as hierarchical models. An hierarchical simulation model is a simulation model that can be represented by levels in an hierarchical composition tree. Figure 1, shown on the next page, illustrates an example of such a composition tree for a manufacturing system. Within the hierarchy, upper levels contain more abstract representations of system components (e.g., plants and departments) than lower levels. Lower levels are typified by components within the real system which are "observable" and/or "touchable" (e.g., machines and material handlers) .

As illustrated in Figure 1, hierarchical simulation models can be formulated using either a top-down or a bottom-up strategy. The top-down strategy is characterized by recursive decomposition of complex, abstract model components into simpler ones. In a manufacturing model this would involve decomposing a plant into departments then

departments into workcenters then workcenters into machines. Conversely, the bottom-up strategy is characterized by recursive grouping of detailed modeling constructs into more complex ones (i.e., grouping machines into workcenters then workcenters into departments then departments into a plant). The grouping of machines into workcenters (or workcenters into departments, etc.) is typically based on either common material handling or common control mechanisms and strategies or both.

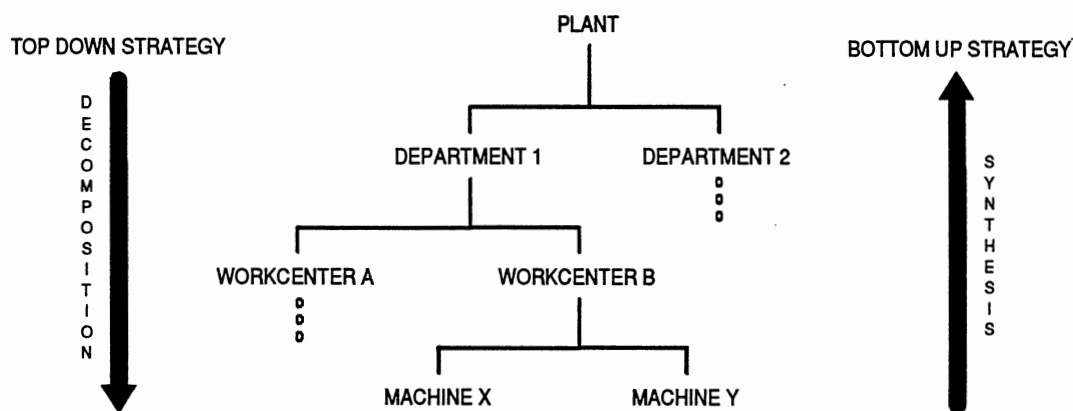


Figure 1. Hierarchical Structure Of A Manufacturing System

Fundamental Issues of Simulation Modeling

To successfully utilize simulation modeling, the modeler must consider three major issues; system boundary, level of abstraction and experimental frame. The system boundary refers to the partitioning of the real system into that part which will be explicitly represented in the model and the remaining part whose impact will be passed as inputs to the model. The level of abstraction of a model is a measure of the detail with

which the components are portrayed within the simulation model. In an hierarchical simulation model the level of abstraction is directly related to the composition tree level at which downward (upward) recursion stops when using the top-down (bottom-up) modeling strategy. Decisions regarding system boundary and level of abstraction are guided by the experimental frame. The experimental frame is a statement of the specific purpose(s) for which the simulation experiment(s) are being conducted, in other words, the experimental frame is determined by the information which is needed for decision making.

Historically, one of the significant disadvantages of simulation modeling is that models are typically constructed as single use models. That is, once used for its original purpose, a particular model is rarely used again. When a new problem is encountered, a new model is generated from scratch even though it may include elements contained in earlier models. The cost of this approach, measured in both dollars and hours, causes many to question the value of using simulation to model large complex systems.

Object Oriented Modeling

The advent of object oriented programming (OOP), a paradigm in which all program variables are represented as "objects", appears to be a significant advancement toward the development of multiple use, general purpose models. OOP achieves this advancement through implementation of four key concepts: encapsulation, message passing, late binding and inheritance. These concepts lead to three major differences between OOP models and procedural language models. First, OOP models are typically structured to more closely parallel the "real world" system being modeled since program objects parallel real world objects. Second, a modeler's ability to understand, modify, and maintain a model are improved since objects incorporate both data and methods of operating on the data into a single coherent entity. Finally, reusability is enhanced

through inheritance and through the use of instances of one class as internal components of other classes.

Using the concepts outlined above, an object oriented modeling (OOM) environment is under development within OSU's Center for Computer Integrated Manufacturing. This environment specifically targets reusability as a key development factor. As a consequence of the reusability emphasis, the bottom-up modeling strategy is employed to create modeling constructs for the lowest level physical, informational, and control components of a real world manufacturing environment. These modeling constructs comprise both generic elements and company-specific elements and are referred to as modeling primitives. After being validated, these primitives become part of a manufacturing modeling database.

If a particular combination of primitives is logically and/or behaviorally related, the collected representation can also be stored in the modeling database. This collected representation is known as a coupled model. Within a manufacturing context, examples of coupled models include (1) a collection of machines and material handling devices forming a workcenter, (2) a collection of workcenters forming a department, and (3) a collection of departments forming a plant. The ability to create and save coupled models can significantly enhance a modeler's productivity. When a simulation model is appropriate as a decision making aid, primitives and coupled models are drawn from the database and linked together, in a bottom-up fashion, to form an overall model of the situation of interest.

Another key consequence of the reusability emphasis is the implementation of separation. The implementation of separation involves the creation of separate and distinct modeling constructs for physical elements, information flows and control decisions. Traditional simulation languages do not provide natural constructs for separately and distinctly modeling physical, information, and control elements. Further, the constructs provided for information and control are frequently hard coded and dispersed into

the model. This results in code that is hard to modify and difficult to use for multiple purposes.

Designing for reusability involves identification of behaviors that are useful in more than one context. In general this implies a system design which adheres rather strictly to the "one component - one function" doctrine. If a component performs more than one of the three basic functions (i.e. physical, information, and/or control/decision), its usage becomes limited to situations in which all of its functions are required. On the other hand if a strict one-to-one functionality is maintained between component and function, then the components truly become "building blocks" (i.e. modeling primitives) from which a total system model can be constructed.

Another advantage of the separation of physical, information, and control objects is that it allows the system modeler to think of these elements independently during model development. This provides a more "natural" modeling environment. When developing the physical model, the modeler need not be concerned with information or control aspects. The process involves selecting the appropriate physical components without being constrained by concerns regarding how to model the information flow. Similarly, information flow is considered without regard to physical objects. This independence facilitates the creation of models with a higher degree of integrity and greater flexibility relative to experimentation with the model.

Top-Down Versus Bottom-Up Modeling

Marked differences exist in the modeling process when the bottom-up strategy is employed. Traditionally, the modeling process proceeds through the following steps:

- o The problem statement is defined.
- o The simulation experimental frame is defined based on the problem statement.
- o The model's system boundary and level of abstraction are determined with respect to the experimental frame.

- o The model is coded, validated, and exercised.
- o The results are presented for decision making.

This is inherently a top-down process since the system boundary and level of abstraction, as well as the model coding and validation, are a function of a specific problem statement and experimental frame.

Within the OOM environment a markedly different modeling process is required.

This process is composed of the following major steps:

- o The modeling database of reusable primitives and coupled models is defined.
- o The problem statement is defined.
- o The experimental frame is defined.
- o The appropriate modeling primitives and coupled models are extracted and the "building blocks" are assembled.
- o The model is exercised.
- o The results are analyzed for decision making.

The fundamental difference between this modeling paradigm and traditional modeling paradigms is that the modeling database is created prior to defining a problem statement and an experimental frame. The building blocks must exist before a particular model can be constructed. The development of this database is not a trivial effort. Its up-front creation (and subsequent maintenance) is a cost in time and resources that must be carefully considered. This approach is clearly more suitable for an enterprise which intends to use OOM capabilities as an integral part of its business operations and not as simply an ad hoc project analysis tool.

The a priori creation of the modeling database gives rise to an interesting question. "If the experimental frame is not known, at what level of abstraction do you create the primitives?" In order to have maximum flexibility, the implication is that modeling primitives must be defined at their lowest possible level so that all behaviors of potential interest are captured. Carried to the extreme, this would imply modeling physical objects

at a level equivalent to atoms and molecules in physics. Although this may be conceivable and perhaps even doable, it is doubtful that it would be of any practical value in a manufacturing simulation model. A more reasonable level is one which captures "observable behaviors" within the primitives. In this context, an observable behavior is one which impacts (or potentially impacts) the performance of the object relative to the goals of the system.

The Impact Of OOM On The Fundamental Issues

The OOM paradigm has a significant impact on the fundamental issues of simulation (system boundary, level of abstraction, and experimental frame). Using the OOM paradigm the system boundary is more fluid. The system boundary can be enlarged by selecting additional modeling components from the database and incorporating them into the model. Similarly, components can be removed from the model to reduce the system boundary.

The flexibility to enlarge or reduce system boundary adds robustness to the simulation experimental frame issue. With the ability to freely move the system boundary, a modeler can deal with a much broader range of experimental frames. By defining modeling primitives at their lowest levels, they embody all potential behaviors of interest. This gives the modeler great depth in the range of experimental frames that can be explored. Thus, both breadth and depth of potential experimental frames are enhanced under OOM.

The model abstraction issue remains a problem under OOM. The problem occurs since all models are built from a collected set of modeling primitives. The problem manifests itself in the execution times of a model. Execution time becomes a problem if the experimental frame encompasses significant breadth and/or depth relative to the primitives in the modeling database. While the modeling database may contain the appropriate modeling elements to construct the system model, the number of elements

involved may result in excessively long model execution times. The problem is one of level of abstraction. Bottom-up modeling within OOM does not provide a convenient mechanism to adjust the level of abstraction based on the experimental frame.

Conceptually, within the composition tree hierarchy, a workcenter is viewed as a more abstract representation than the collection of machines and material handlers from which it was built. However, within OOM it remains a collection of primitive objects whose behaviors are modeled at low levels. The workcenter designation (within OOM) allows certain global controls to be placed over the set of primitives but the behaviors are still detailed. Even the coupled model of the workcenter (if one was built and saved) is in actuality nothing more than a convenient naming convention which is used to easily refer to the collected set rather than the individual members.

Addressing The Abstraction Issue: Metamodeling

A base model is a model whose component parts are all modeling primitives. All models developed within the OOM environment presented above are base models. The major question resulting from the previous section is whether a more abstract representation of some portion of a base model can be implemented to improve model efficiency while retaining the detail and accuracy of the performance measures required by the experimental frame being investigated.

If a more abstract representation is used for a workcenter, it in essence becomes a "model of a model." Within the relevant literature (see Chapter 3) many terms have been coined for such a representation. Among the terms encountered are aggregate model, simplified model, metamodel, reduced model, lumped model, flow-equivalent model, auxiliary model, and repro-model. The term of choice for this research is metamodel. A metamodel is a "model of a model" according to Schriber [1987] who used the term in suggesting the use of regression equations in conjunction with simulation. Since the

methodology to be used within this research is consistent with Schriber's suggestion, his term for the abstraction will be used.

Within the context of this research, metamodels are used in concert with detailed simulation models. The metamodel transparently replaces (hopefully in a modular, plug compatible fashion) the corresponding portion of the detailed model. The intent of the metamodeling methodology developed in this effort is that the metamodel be more computationally efficient than the detailed simulation and yet deliver approximately equivalent steady state behavior across the performance measures of interest. If analysis of transient behavior is the goal of a simulation study, then metamodels of the type considered in this research are not appropriate.

The metamodel replacement process is depicted in Figure 2 and Figure 3 on the next page. Figure 2 is the base model of a hierarchical manufacturing plant (depicted in OOM separation format). Note the various machine primitives, material handling primitives, information primitives, and control primitives throughout the plant specification. Figure 3 shows the same plant model with a metamodel in place for Workcenter 2. Note that for the metamodeled component, the detailed system elements have been replaced by a single component whose aggregate behavior is represented as a sampling distribution.

The metamodeling question must be asked anew each time a base model is used to investigate a new experimental frame. The base model must be examined in light of the experimental frame to determine which workcenters, if any, are amenable to a more abstract representation without significantly degrading the information content of the experimental results. From a benefit/cost perspective, the primary benefit sought is a gain in model execution speed and the cost incurred is an aggregate behavior rather than a collection of detailed ones. This process of a single base model spawning multiple metamodel versions is depicted in Figure 4 on page 21.

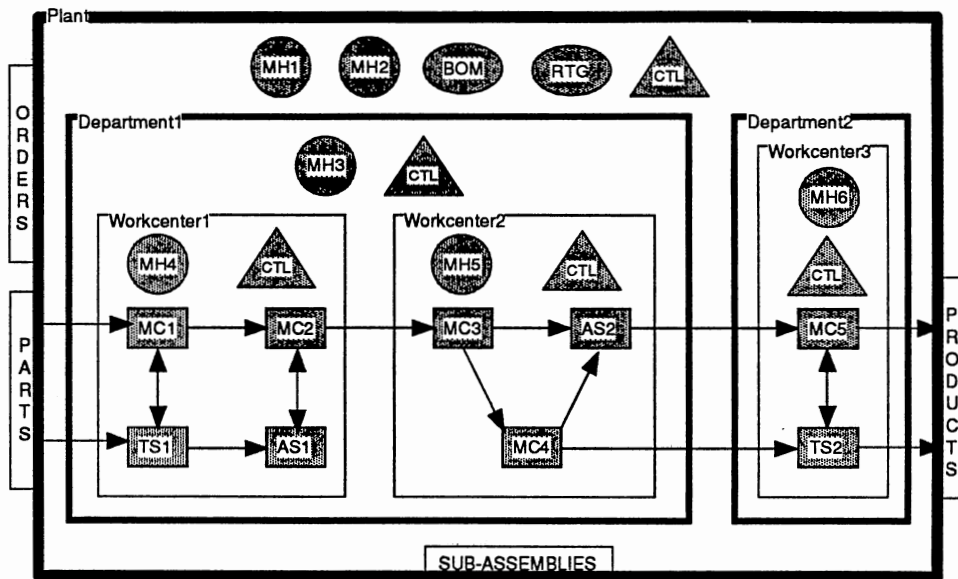


Figure 2. Base Model Of A Hierarchical Manufacturing System

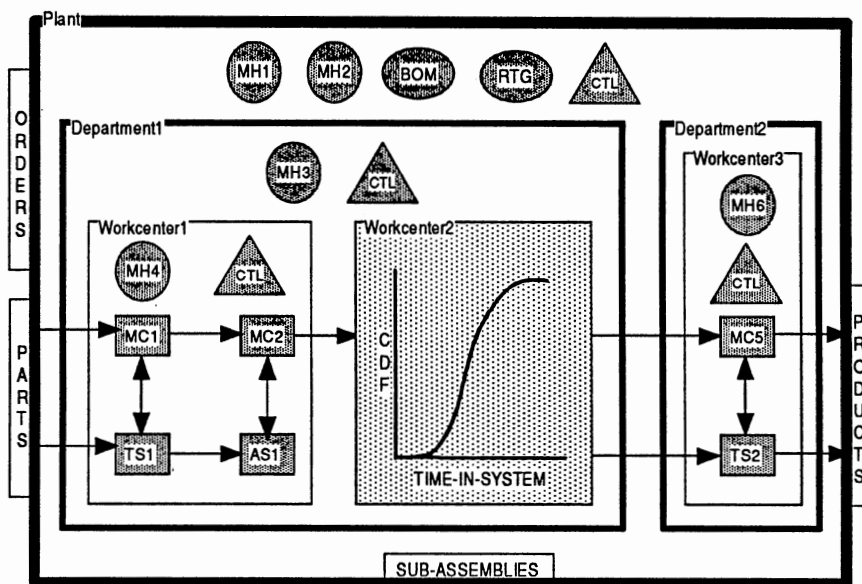


Figure 3. Metamodel Version Of A Hierarchical Manufacturing System

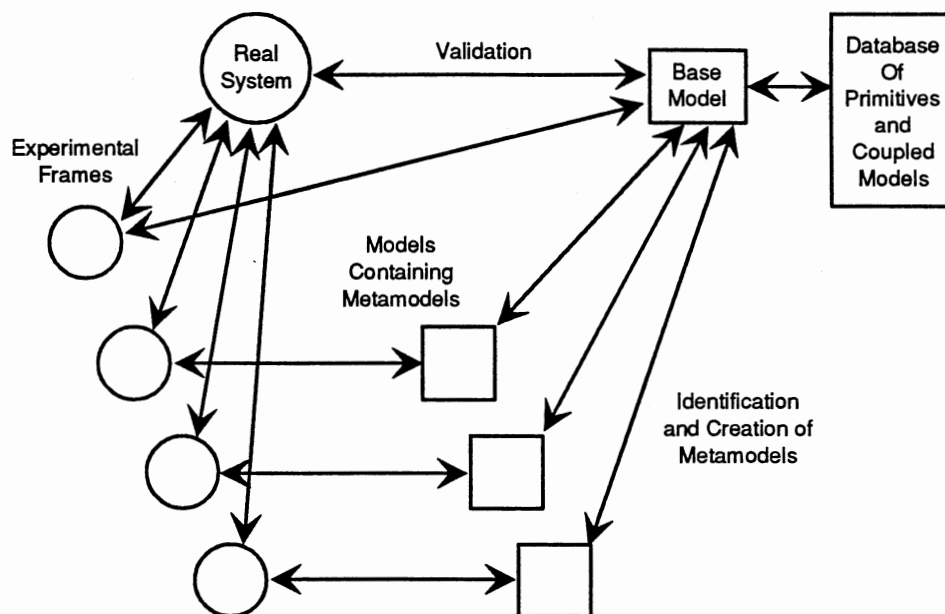


Figure 4. Metamodels And Experimental Frames

A metamodel could potentially take many forms. Any function which maps an input variable(s) to an output variable(s) is a potential candidate. Two classes of metamodels will be considered in this research: observation-based metamodels and queueing network metamodels. An observation-based metamodel is a metamodel which is created by monitoring and recording the behavior of the base model over time and then attempting to discover an analytical function which mimics the relationship between the input variable and the performance measure. A queueing network metamodel is a metamodel which is created by formulating the base model as a queueing network and then, using the relevant theory or published solutions, solving the network for the desired relationships.

Hybrid Modeling

As stated above metamodels are used in concert with detailed simulation models. By introducing a metamodel the simulation is in effect running at two levels of abstraction simultaneously. This fact by itself is not noteworthy. In many, if not most, models developed using the top-down strategy, multiple levels of abstraction are represented (sometimes by choice, sometimes by default). Detailed model components are developed where needed for the experimental frame while the remaining components are left at higher levels of abstraction. Modeling of this type is frequently referred to as multi-level or mixed-level modeling.

The noteworthy differences about the approach pursued in this research are twofold. First, the approach seeks true "plug compatibility" between the metamodel and the base model. The modeler should be able to save both versions in the modeling database and freely chose the metamodel or the base model depending on the experimental objectives. The system model into which either of these is fitted should be totally indifferent to the form of representation.

The second noteworthy difference is that the approach pursued here represents true hybrid modeling rather than multi- or mixed-level modeling. Within the context of this research, a hybrid model is a model within which simulation techniques are used in conjunction with analytical expressions to represent the behavior of a real or proposed system. The functional form of the analytical expressions are determined through either the observation of a detailed model or the solution of a queueing network.

Unanswered Questions

The above development leaves unanswered many questions regarding hybrid metamodeling of hierarchical manufacturing systems. Among the unanswered questions are:

- o What coupled models within a hierarchical manufacturing model are amenable to the creation of a "plug compatible" metamodel?
- o Can the structure, inputs, and outputs of the coupled model be used to suggest/select an appropriate metamodel?
- o Can a rule based decision process be used to suggest when a metamodel is appropriate given a set of coupled models and an experimental frame?
- o Can a valid and robust metamodel be developed by observing the behavior of a base model over a range of input values?

This research endeavors to address these questions and gain insight into a methodology for answering them.

CHAPTER III

BACKGROUND OF THE STUDY

Introduction

This chapter contains a formal review of the literature related to hybrid meta-modeling of hierarchical manufacturing systems. From a breadth and depth standpoint, much of the knowledge gained from the literature below is contained in the preceding (Problem Statement) chapter. The review here will be broken into three broad categories: metamodeling of hierarchical systems, hybrid modeling, and performance modeling of manufacturing systems. Considering each area independently, the amount of literature available ranges from quite small, for metamodeling, to rather large, for performance modeling. Significantly smaller bodies of literature are available when combinations of the areas are considered. These combinations will be discussed within the area that, in the author's opinion, they make the greatest contribution. Only one major reference, Sevinc [1988], was found that simultaneously addresses all three areas. Significant differences exist between Sevinc's work and this research. These differences are elaborated below.

Metamodeling Of Hierarchical Systems

A metamodel is "model of a model." The concept of metamodeling first appeared in the literature in the early 1970s. Meisel and Collins [1973] introduced the term re-pro-modeling as an approach to reductive modeling. They attempt to obtain a model of a complex relationship that allows convenient and repeated use within a larger (e.g.,

hierarchical) framework. They define repro-modeling as a process for developing an approximation to, or condensation of, a complex computer-based model. The repro-model is designed to give the same results as a complex series of models to an appropriate level of accuracy. The advantages cited for repro-modeling are that it overcomes many of the impediments to the broad use of complex models: computational costs, excessive input requirements, and difficulty in interpretation of the implications of a model.

Meisel and Collins propose two functional forms for repro-models. They are composed functions and continuous piecewise linear functions. Of the two, continuous piecewise linear functions are preferred because they are easier to interpret both qualitatively and quantitatively and usually extrapolate well beyond the "region of approximation." The concepts of repro-modeling have been applied to environmental quality, traffic flow, and radar targeting of complex objects.

The term metamodel per se was first introduced by Blanning [1975]. Blanning's work is focused exclusively on using metamodels for post-simulation sensitivity analysis. Data collected from a series of simulation runs is used to construct another model to yield sensitivity information over a very narrow range. In this context, the metamodel is not used within a larger framework but as a stand alone sensitivity analysis tool. The functional form for the metamodel is case specific. For cases where the form is differentiable, Blanning provides a methodology for evaluating and interpreting the partial derivative matrices.

Blanning observes that no theory exists to suggest how a metamodel should be constructed. Therefore, metamodels must be constructed on an ad hoc basis. The two recommended ad hoc techniques for metamodel construction are to infer a functional form from observation-based data and to construct a simple analytical model. Blanning's conclusion is that while metamodels appear powerful and useful, there is not sufficient experience to suggest when and how they should be constructed and used.

Kleijnen [1979] extends the two works cited above with regression-based methodology. The primary advantage of this approach is that the metamodels are linear in their parameters and thus easy to interpret. An additional advantage is that the statistical basis for regression analysis is well formed and allows quantitative measures of the accuracy of the metamodel.

Beyond the metamodeling advantages cited by other authors, Kleijnen identifies "selling" of results as a significant benefit. This advantage derives from the fact that simple metamodels (particularly regression models) are easier to understand for non-technical users. Besides being easier to understand, Kleijnen cites Geoffrion [1976] in claiming that a higher purpose is served (i.e., "the purpose of modeling is insight, not numbers").

Two notable extensions to Kleijnen's work should be mentioned. Friedman [1984] extended the general linear metamodel to include multiple response variables. Her technique is known as the multivariate general linear metamodel. She validates her method using an analysis of the mean performance of an M/M/1 queue. The second notable extension occurred when Friedman extended her own work with canonical correlation analysis [Friedman 1986]. This technique is a multivariate technique that is useful for identifying relationships between sets of variables. In the metamodeling context, it aids in the unmasking of complex relationships and interdependencies between system control variables and response variables.

The most recent reference to metamodeling per se is found in Schriber [1987]. While Schriber does not cite any of the above metamodeling works, he takes the strong position that regression equation metamodels should be applied more frequently. His rationale in calling for greater use of these techniques is that they "best explain the behavior of the performance variables."

Zeigler [1984] refers to the metamodeling concept as "aggregation" or "simplification" within his DEVS formalism. The DEVS formalism is a hierarchical

bottom-up modeling formalism specifically designed for discrete event simulation. In Zeigler's terminology, the application of aggregation to a "base" model results in a "lumped" model. A valid aggregation is one in which the lumped model is equivalent to the base model within the experimental frame of interest.

Zeigler presents four general forms of simplification procedures. The one relevant to this research effort is a mapping procedure that maintains compatibility between the base and lumped models at the input-output level. This is nothing more than maintaining "plug-compatibility" within the hierarchical model. Two approaches are proposed to solve for the lumped model. If the system is tractable a complete solution may be expressed in analytical form. Otherwise numerical simulation is used to build tables to estimate the response function.

Sevinc [1988] and Zeigler [1990] operationalize the simplification procedure proposed by Zeigler. While this work looks promising with regard to the DEVS formalism and the structural form of simplified models, its behavioral performance is weak. Zeigler's DEVS formalism is the foundation for the research. A DEVS model is a state machine whose transitions are specified by the external events happening at its inputs, by its internal activities and by its states. Sevinc identifies two critical issues in model simplification: (1) the reduction of the complexity of the model in terms of (computer) memory and time and (2) to ensure that the model and its simplified version behave closely given a goodness of fit criterion within an experimental frame of interest.

Sevinc's work focuses mainly on the reduction of model complexity. The reduction is accomplished through model "observers" that are attached to the base model. The observers monitor model states and transitions as the simulation progresses. These states are "lumped" into pairs (phase and time left) that maintain consistent I/O behavior with the base model. The simplified model is constructed by accumulating these "lumped" pairs and calculating transition probabilities from observed data.

This simplification process is demonstrated using a simulation of a local area network model with various numbers of nodes and two different access protocols. Sevinc's results demonstrated that (1) the simplification process using observers and lumped models is totally compatible with the DEVS formalism and (2) the run time for the base model increases very rapidly whereas the run time for the lumped model increases very slowly as the number of nodes grows.

Behaviorally, Sevinc's lumped models are less encouraging. Under one access protocol, the performance accuracy (measured by the relative number of packets transmitted without collision) ranged from 90% to 40%, under the other protocol, 100% to 25%. Also, in the simplification process, knowledge of the packet turnaround times is lost.

The most important differences between the current proposed research and Sevinc's research are:

- o Sevinc places heavy emphasis on compatibility with the DEVS formalism; no emphasis is placed on DEVS (or any other formalism) in the current effort.
- o Sevinc places more emphasis on reduction of complexity than on behavioral performance; this research strives to do exactly the opposite.
- o Sevinc's research was generic within the DEVS structure; this research will seek to take advantage of domain specific behavior (i.e., workcenters within a hierarchical manufacturing model).

Hybrid Modeling

Hybrid simulation methodology combines both analytical methods and simulation methods. The goal of hybrid modeling is to define models that are both more robust than pure analytical models and less computationally complex than pure simulation models. In this research, hybrid modeling encompasses not only analytical models working in

concert with simulation, but also observation based functions (e.g., Zeigler's lumped models).

Like most other techniques, there are advantages and disadvantages associated with using hybrid modeling. The advantages are:

- o It exploits the benefits of both analytical and simulation modeling.
- o It reduces the computational complexity of the model.
- o It frequently leads to variance reduction in performance measure estimates.

The disadvantages are:

- o A modeler must have knowledge of both analytical (i.e., metamodel) and simulation techniques.
- o Interfacing the analytical (i.e., metamodel) and the simulation models can be difficult.

In general, hybrid models can be thought of as a subset of the set of system simulation models and a superset of the set of mathematical models of systems. A system that is amenable to hybrid modeling must be decomposable from an input/output perspective. The hierarchical manufacturing systems defined within this research fall within this category.

Schwetman [1977; 1978] introduced the concept of hybrid simulation to solve a class of computer system models. His results demonstrated a significant reduction in computational expense while maintaining similar results for performance variables. Even in cases where the decomposability assumption was not completely met, hybrid modeling still proved valuable in narrowing the range of alternatives. Chiu and Chow [1978], Thomasian and Gargeya [1985], and O'Reilly and Hammond [1984] have also applied hybrid simulation to computer system models and local area networks.

Shanthikumar and Sargent [1983] present a unifying definition of hybrid modeling. They define four classes of hybrid models by differentiating the way in which the simulation and analytical models interact. A Class I model is a model whose

behavior over time can be completely decomposed into two independent parts, one analytic and one simulation. The simulation part of the model can be carried out without intermediate interaction with the analytic part and vice versa. If the simulation model and the analytic model run in parallel with intermediate interaction, then the model is Class II.

Class III and IV models use analytic models and simulation models, respectively, as models of the total system. In a Class III model, a simulation model is used in a subordinate way to provide estimates for at least some values of the parameters in the analytic model. Class IV models obtain values for some or all of the simulation model parameters from the solution procedure of an analytic model. Usually the results of the analytic model are generated and stored for use by the simulation model rather than continuously executing the analytic model as the simulation model moves through time. The models to be considered in this research effort are Class IV.

Many authors have cited applications of hybrid modeling to manufacturing systems in the literature. Among them are Tolopka and Schwetman [1979], Dietrich and March [1985], Nyman [1987], and Haider, Noller, and Robey [1986]. Unhappily, in each of these cases the hybrid model was either Class I or Class II rather than Class IV which is of interest to this research.

Performance Modeling Of Manufacturing Systems¹

Performance evaluation models determine the performance measures such as throughput or production rates, equipment utilization, work in process levels, and part flow times that can be expected to result from a given set of decisions. These decisions are typically expressed in terms of products to be manufactured, number and type of machines, routings and operation sequences, number and type of material handlers, and

¹The majority of the material in this section was originally developed as part of (1) an NSF proposal on which the author was an associate investigator [Mize, Kamath, and Leemis 1990] and (2) a technical paper co-authored by the author [Basnet et al., Object-Oriented Modeling, 1990].

capacities of WIP areas. Four techniques have been used extensively for performance evaluation of manufacturing systems. These techniques are discrete event simulation, queueing networks, fast simulation, and Petri nets. The following sections briefly review these techniques and their application. Broader reviews of performance evaluation techniques for manufacturing systems are contained in Buzacott and Shanthikumar [1980], Buzacott and Yao [1986], Segal and Whitt [1989], and Suri, Sanders, and Kamath [1990].

Discrete Event Simulation

Discrete event simulation is the only viable approach to the detailed performance analysis of complex manufacturing systems. Simulation models can mimic the operation of a system in as much detail as required by the user and therefore can be very accurate. Compared to other performance evaluation techniques, assumptions required in a simulation model are closer to reality. Also, this is one of the few methodologies that can be used for studying both transient and steady state behavior. Unfortunately, simulation modeling is often a time consuming task with computationally expensive analysis.

The history of simulation modeling can be broken into five periods: the era of custom programs, the emergence of simulation specific languages, the second generation of simulation languages, the era of extended features, and the current period [Nance 1984]. Early simulation modeling was performed using custom programs written in general purpose computer languages, such as FORTRAN. Although this approach proved the viability of simulation modeling, the models were typically expensive and time consuming to design and maintain. Usually, the work done on a specific modeling project could not be easily used during subsequent modeling efforts, even when many modeling elements overlapped. This resulted in simulation being used primarily on large expensive projects.

In the early 1960s, as the field of simulation developed further, discrete event simulation languages such as GPSS, SIMSCRIPT, GASP, and SIMULA appeared [Mitrani 1982; Nance 1984]. These languages were primarily written in general purpose languages but provided generic functions and subroutines to perform many of the tasks routinely required in simulation, such as calendar functions and statistics collection. Unfortunately, the bulk of simulation model development effort was still spent in developing problem specific code with little reusability. In the late 1960s, a second generation of simulation languages emerged. In most cases, these languages were more powerful replacements of their predecessors (e.g., GPSS V, SIMULA 67, and GASP IIA).

In the 1970s, as the use of simulation modeling grew, developments in simulation languages were driven toward the extension of simulation specific languages to facilitate easier and more efficient methods of model translation and representation. Many languages that evolved from these developments, GPSS, SLAM II, SIMSCRIPT II, and SIMAN, are still widely and actively used today [Law and Haider 1989; Pegden 1986; Pritsker 1986].

In the early 1980s, many changes were occurring in the computer hardware arena; personal computers were becoming a mainstay, high resolution graphics and animation were efficiently realizable, and artificial intelligence (AI) and expert systems were seeing a resurgence with practical implementations [Nilsson 1980]. These changes had, and continue to have, a direct impact on simulation methodologies. Simulation modeling is now open to a much broader base of potential users through advances such as: menu and icon driven model builders, expert systems to aid in the building and debugging of models, graphs and charts to display model results both during and after execution, and model animation to view the operation of the system as a whole or to zoom in on a specific area of interest.

In the area of graphics and animation, packages such as SLAMSYSTEM, Cinema/SIMAN, and SIMFACTORY [Nance 1984] are among the leading edge competitors. The animation and graphics are typically developed and presented as an integral part of the simulation language. By contrast, AI and expert system concepts impact simulation modeling through a simulation "front-end" or application generators. These tools interact with the user and ultimately result in a set of code that can be passed directly to the simulation language. Among the leading edge competitors in this area are EZSIM [Khoshnevis and Chen 1987], SMP [Endesfelder and Tempelmeier 1987], and MAGEST [Oren and Aytac 1985].

Queueing Networks

Queueing network models have emerged as one of the most widely studied analytical models of modern manufacturing systems [Buzacott and Shanthikumar 1980; Buzacott and Yao 1986; Solberg 1977; Suri, Sanders, and Kamath 1992]. A node in a queueing network model generally represents a workstation in the manufacturing system. If the workstation consists of a single machine then the queueing node has a single server. Multiple identical machines result in a multiple-server node. The entities flowing through the queueing network represent the parts moving through the manufacturing workstation. Preceding each node is a queue whose capacity is dictated by the number of parts that are allowed to wait for service in front of the workstation.

If the number of entities within the queueing network is fixed then the network is classified as closed, if the number fluctuates it is open. At a minimum, the specification of a queueing network model must include:

- o the service time distribution for each node
- o the routing sequence of parts within the network
- o the arrival process for parts (open network) or the number of parts in the network (closed network)

Generally, the service time and routing information is obtained from the process plans of the manufacturing system. Techniques for transforming the process plans into queueing network form have been demonstrated by Bitran and Tirupati [1988] and Whitt [1983].

One popular and heavily used type of queueing network model is the Jackson-Gordon-Newell type [Conway and Georganas 1989; Gelenbe and Pujolle 1987; Gordon and Newell 1967; Jackson 1957, 1963; Kelly 1976; Suri 1983; Suri and Hildebrandt 1984]. These networks, commonly called Jackson networks, possess a product form steady state distribution. The name product form derives from the fact that the steady state distribution of the joint probabilities of the lengths of the queues is written as a product of the marginal distributions for each queue.

Jackson networks form the basis for the Product Form Analysis (PFA) method. Examples of the PFA method are techniques such as CAN-Q [Solberg 1977] and Mean-Value Analysis of Queues (MVAQ) [Reiser and Lavenberg 1980; Suri and Hildebrandt 1984]. These techniques are used to compute throughput and utilizations in closed networks. For certain flexible manufacturing system (FMS) applications performance analysis using the PFA method has been successful [Buzacott and Yao 1986; Solberg 1977; Suri and Hildebrandt 1984] although the product form assumptions are often restrictive.

Baskett et al. [1975] expanded the range of product form solutions by introducing what are commonly called BCMP networks. BCMP networks retain a product form solution but allow for multiple classes of customers and service disciplines other than first-come-first-serve. Agrawal [1985] has detailed a list of assumptions required for a product form solution to be calculable. This list includes:

- o One Step Behavior: A state transition can occur only due to the departure of a single customer from one resource to another or outside the system, or due to the arrival of a customer from the outside,

- o Flow Balance: The number of arrivals (in each class) at a device must equal the number of departures (in each class) from the device,
- o Device Homogeneity: A device's service rate for a particular class does not depend on the state of the system in any way except the total device queue length and the designated class's queue length. This assumption implies:
 - Single Resource Possession: A customer may not be present (waiting for service or receiving service) at two or more devices at the same time;
 - No Blocking: A device renders service whenever customers are present, i.e., its ability to render service is not controlled by any other device;
 - Independent Customer Behavior: Interaction among customers is limited to queueing for physical devices, e.g., there should not be any synchronization requirements;
 - Local Information: A device's service rate depends only on local queue length and not on the state of the rest of the system; and
 - Fair Service: If service rates differ by class, the service rate for a class depends only on the queue length of that class at the device and not on the queue lengths of other classes. This means that the server does not discriminate against customers in a class depending on the queue lengths of other classes.
- o Routing Homogeneity: The customer routing should be state independent.

Baskett et al. [1975] have shown that these assumptions are met if a workstation satisfies one of the following conditions:

- o The service discipline is first-come-first-serve, all customers have the same service time distribution at this station, and the service time distribution is exponential. The service rate can be state dependent based on the number of customers at the station.

- o There is a single server at a service station, the service discipline is processor sharing² and each class of customer may have a distinct service time distribution. The service time distributions must have rational Laplace transforms.
- o The number of servers in the workstation is greater than or equal to the maximum number of customers that can be held at this station and each class of customer may have a distinct service time distribution. The service time distributions must have rational Laplace transforms.
- o There is a single server at a workstation, the queueing discipline is "preemptive resume last-come-first-serve"³, and each class of customer may have a distinct service time distribution. The service time distributions must have rational Laplace transforms.

To illustrate the application of PFA, two simple queueing network examples are given below. The first example is an open Jackson queueing network. The manufacturing system is a two stage tandem line with Poisson arrivals and exponential service rates. This network is illustrated in Figure 5 on the next page. For this network to be stable, λ must be less than μ_1 and μ_2 . The state of this system can be defined in terms of the number of customers at each station (n_1, n_2). This represents a continuous time Markov chain.

If P_{n_1, n_2} is defined as the probability of being in state (n_1, n_2), then the balance equations (i.e, rate the process leaves a state = rate the process enters the state) for this system are given in Table I on the next page. Using these balance equations along with the fact that the P_{n_1, n_2} 's must sum to one, it can be shown that a solution to the balance equations is given by:

²For example, when there are n customers at the station, each customer is receiving $1/n$ minutes of service per minute.

³New customers have absolute priority, that is, a newly arrived customer interrupts on-going service to start its own. The interrupted customer is placed at the head of the queue and re-starts service from the point it was interrupted when the customer causing the interruption finishes.

$$P_{n1,n2} = \left(\frac{\lambda}{\mu_1} \right)^{n1} \left(1 - \frac{\lambda}{\mu_1} \right) \left(\frac{\lambda}{\mu_2} \right)^{n2} \left(1 - \frac{\lambda}{\mu_2} \right)$$

In the general case, this result can be extended to an m-stage tandem line where

$$P_{n1,n2,\dots,nm} = \prod_{i=1}^m \left(\frac{\lambda}{\mu_i} \right)^{ni} \left(1 - \frac{\lambda}{\mu_i} \right)$$

From these results, other performance measures of interest in a manufacturing system, such as average number in system and average waiting time, can be calculated.

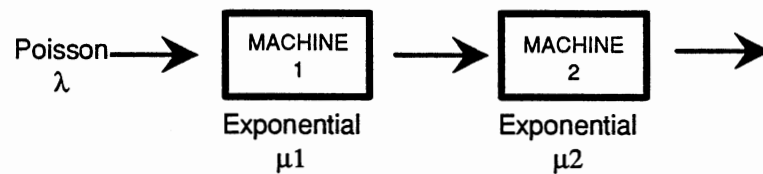


Figure 5. A Simple Open Jackson Queueing Network

TABLE I
BALANCE EQUATIONS FOR FIGURE 5

STATES	BALANCE EQUATIONS
0,0	$\lambda P_{0,0} = \mu_2 P_{0,1}$
$n1,0; n1 > 0$	$(\lambda + \mu_1) P_{n1,0} = \mu_2 P_{n1,1} + \lambda P_{n1-1,0}$
$0,n2; n2 > 0$	$(\lambda + \mu_2) P_{0,n2} = \mu_2 P_{0,n2+1} + \mu_1 P_{1,n2-1}$
$n1,n2; n1 > 0, n2 > 0$	$(\lambda + \mu_1 + \mu_2) P_{n1,n2} = \mu_2 P_{n1,n2+1} + \mu_1 P_{n1+1,n2-1} + \lambda P_{n-1,n2}$

The second example is a closed Jackson network. Recall that a closed network is one that has a fixed number of customers within the system. A simple closed Jackson network comprising an inspection and repair station is illustrated in Figure 6. The flow balance equations for this network are given in Table II on the next page. The solution to these balance equations takes the form:

$$P_{n_1, n_2} = \frac{1}{C(N)} \rho_1^{n_1} \rho_2^{n_2}$$

where:

$$\rho_1 = k/\mu_1$$

$$\rho_2 = kp_2/\mu_2$$

k is an arbitrary constant

$C(N)$ is a normalizing constant.

P_{n_1, n_2} is independent of the selection of the constant k , but ρ_1 , ρ_2 , and $C(N)$ are all functions of k .

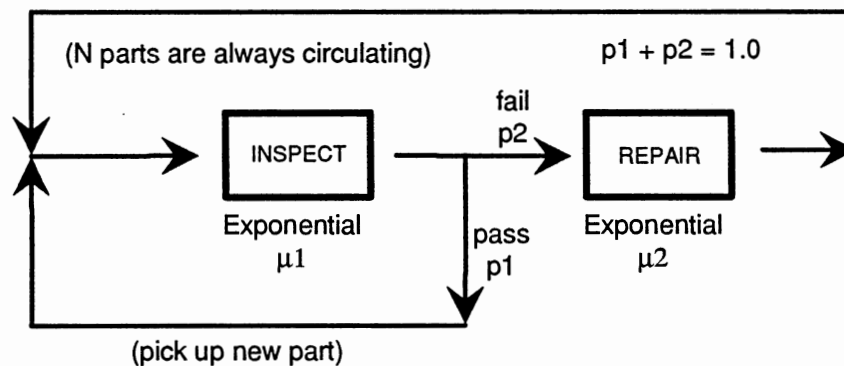


Figure 6. A Simple Closed Jackson Queueing Network

TABLE II
BALANCE EQUATIONS FOR FIGURE 6

STATES	BALANCE EQUATIONS
N,0	$\mu_1 P_2 P_{N,0} = \mu_2 P_{N-1,1}$
0,N	$\mu_2 P_{0,N} = \mu_1 P_2 P_{1,N-1}$
$n_1, n_2; n_1 > 0, n_2 > 0$	$(\mu_1 P_2 + \mu_2) P_{n_1, n_2} = \mu_2 P_{n_1-1, n_2+1} + \mu_1 P_2 P_{n_1+1, n_2-1}$

If k assumes the value μ_1 , then $\rho_1 = 1$ and $\rho_2 = \mu_1 P_2 / \mu_2$. Substituting these values and combining with the fact that the P_{n_1, n_2} 's must sum to one, it can be shown that:

$$C(N) = \begin{cases} \frac{1 - \rho_2^{N+1}}{1 - \rho_2} & \text{for } \rho_2 \neq 1 \\ N+1 & \text{for } \rho_2 = 1 \end{cases}$$

From these results, other performance measures of interest in a manufacturing system, such as stage utilizations and average throughputs, can be calculated.

For the simple two machine closed network above, the calculation of $C(N)$ is tractable. In the general case, however, the calculation frequently presents computational problems [Ross 1989, 371]. These problems arise due to the combinatorial growth of the number of terms in the summation as the number of machines and/or the number of parts increase. In many of these cases, calculation of mean values for performance measures is of primary concern rather than the joint distribution for the state space P . In these cases, the mean value analysis techniques of Reiser and Lavenberg [1980] and Suri and Hildebrant [1984] are appropriate. These recursive techniques do not require the a priori calculation of $C(N)$ and hence avoid the computational difficulties. Detailed discussion of these approaches is beyond the scope of this effort.

The popularity of product form queueing network models can be attributed to the fact that a relatively well developed theory exists for analyzing such networks.

Furthermore, the performance measures for such networks can be computed using efficient algorithms [Agrawal 1985; Conway and Georganas 1989; Gelenbe and Pujolle 1987], and the models are robust in practical situations [Suri 1983]. The assumption of exponential service times is usually not satisfied by production systems such as an FMS in which the machine times are frequently known quite accurately. Another example is an automatic assembly station in which the assembly time is typically fixed except when a station failure (i.e., "jam") occurs and a repair process is required. In such cases the exponential service time models often do not represent reality faithfully, and the analysis using an exponential assumption can be misleading [Kamath and Sanders 1987]. Furthermore, empirical observation reveals that service time distributions rarely take the form of an exponential function.

For the analysis of large complex systems, the decomposition and aggregation technique developed by Simon and Ando [1961] has proven useful. The primary feature of the technique is to reduce the analysis of a large system into that of a set of smaller, less complex problems through the use of flow-equivalent models. In general, the decomposition and aggregation approach yields approximate results, however, in the special case of a product form network, the results are exact.

Norton's Theorem, developed by Chandy, Herzog, and Woo [1975], is a particular implementation of decomposition and aggregation for constructing an exact reduced system around an arbitrary set of nodes in a product form queueing network. The reduced system is constructed by replacing the set of nodes with a single server flow-equivalent queue. The service rate of the flow-equivalent queue is calculated based on the service rates and routing probabilities of the nodes forming the set. The flow-equivalent queue preserves the equilibrium distribution of the number of parts in the system and hence the mean performance measures.

Agrawal [1985] also reviews many queueing approximations that can be used for non-product form networks. Unfortunately these techniques do not, in general, preserve the distribution of waiting times for work flow items (parts). Additionally, these techniques typically deal with a single non-product form characteristic, whereas, a typical manufacturing workcenter is likely to have multiple characteristics causing it to fall outside the realm of queueing approximations.

In the last few years, there has been considerable interest in approximations to queues with general arrival and service distributions [Kraemer and Langenbach-Belz 1976; Shanthikumar and Sargent 1980; Suri, Sanders, and Kamath 1992; Whitt 1983; Wolff 1989], that require only the mean and variance of interarrival and service times. These techniques are commonly called two moment approximations. Based on these approximations, analysis techniques have been developed for non-exponential networks [Kuehn 1979; Labetoulle and Pujolle 1980; Whitt 1983; Whitt 1984]. Some recently developed tools and techniques for analyzing manufacturing and assembly systems use these two moment approximations as building blocks for more complex models [Bitran and Tirupati 1988; Buzacott and Shanthikumar 1980; Kamath and Sanders 1987; Kamath, Suri, and Sanders 1988; Kamath 1989; Kamath 1991; Kamath and Sanders 1991; Segal and Whitt 1989]. In general, these techniques perform well at high values of ρ , the workstation utilization.

The ability to handle general service times has given these techniques the flexibility to model, although approximately, a variety of situations that commonly occur in production systems such as, different product types, rework and scrap, changing lot sizes, batch service, machine breakdown and repair, deterministic and probabilistic routing, etc. [Bitran and Tirupati 1988; Segal and Whitt 1989]. Some queueing software packages based on these two moment approximations are currently available, for example, MANUPLAN [Suri, Diehl, and Dean 1986; Suri and Diehl 1987; Suri 1988b] and QNA [Segal and Whitt 1989; Whitt 1983].

Fast Simulation

The most popular form of discrete event simulation is the event scheduling approach [Kreutzer 1986; Law and Kelton 1991]. The realization of the simulation model involves the scheduling and execution of events on an event calendar. The overhead load of the simulation is heavily influenced by the continual manipulation of the time-ordered event calendar. Chen and Chen [1990] have demonstrated an approach that eliminates the event calendar for certain classes of queueing network models. This approach is known as fast simulation.

Chen and Chen have shown that for finite buffer, first-come-first serve, single-server tandem queueing systems, fast simulation based on recursion may save up to 80% of run time in estimating certain system performance measures compared to event scheduling simulation. The recursion equations, shown in Table III on the following page, contain relationships among variables such as customer arrival time and start and finish time of service activities. After executing the fast simulation (i.e., generating random variates for the P_{ij} and A_i values and solving the recursion equations), performance measures such as utilization of nodes, waiting time of customers, waiting time at nodes, throughput rates, and time-in-system for customers can be calculated. It should be noted that the recursion equations do not involve any approximations. Fast simulations should yield results identical with traditional simulation.

Kamath, Bhuskute, and Duse [1991] have extended the above approach to consider first-come-first-served service with parallel servers. In contrast to the single server case, the ordering of customers can change at a parallel server node. To accommodate this the customer sequence is recalculated after each parallel server node. Extensions to include

splitting and merging of customer flows, alternate routings, and different queue disciplines have been proposed.

TABLE III
TANDEM NETWORK FAST SIMULATION
RECURSION EQUATIONS

VARIABLES
M = number of nodes; $i = 1, 2, \dots, M$
N = number of customers; $j = 1, 2, \dots, N$
P_{ij} = service time for customer j at node i
A_j = arrival time for customer j at node 1
s_{ij} = starting time for customer j at node i
d_{ij} = departure time for customer j at node i
RECURSION EQUATIONS
$d_{ij} = s_{ij} + P_{ij}$
$s_{ij} = \max(d_{i,j-1}, d_{i-1,j})$
$d_{ij} = \max(d_{i,j-1}, d_{i-1,j}) + P_{ij}$

Petri Nets

A Petri net is a formal graph based model for the description and analysis of systems that exhibit asynchronous, concurrent behavior [Kamath and Viswanadham 1986; Murata 1989; Peterson 1977; Peterson 1981]. Petri nets serve as a natural representation of the flow of information and control in such systems. Several tutorial articles have appeared in the literature [Murata 1989; Peterson 1977]. The wide array of Petri net application areas include computer networks, multi-processing and distributed processing systems, and modern manufacturing systems.

The main advantages of modeling with Petri nets are (1) Petri nets can model a system hierarchically; large complex systems can be represented in a top-down fashion at various levels of abstraction and detail, (2) a systematic and complete qualitative analysis of the system is possible by well developed Petri net analysis techniques, (3) the existence of well formulated schemes for Petri net model synthesis, and (4) performance evaluation using the class of nets known as timed Petri nets. The two most significant disadvantages of Petri nets are (1) analysis techniques for the general case tend to be qualitative rather than quantitative and (2) the state space for complex systems can quickly become too large for analysis. For these reasons, many practical applications use Petri nets with special restrictions that keep the problem tractable.

Petri nets are bipartite directed graphs. The standard Petri net model is defined by a set of places, a set of transitions, and a set of directed arcs that connect places to transitions or vice versa. Places may contain tokens. The marking of a Petri net specifies the location and counts of the tokens in the places of the net. A marking represents a state of the system being modeled. In general the places of the net represent conditions and the transitions represent events. The dynamic behavior of the system is modeled by the occurrence of events (i.e., the firing of transitions). When transitions fire, tokens move from the input places of the transition to the output places resulting in a new marking of the net and thus a new system state. In a Petri net model of a manufacturing system, tokens may represent parts or machines, and places may serve as buffers or machine states. In this manner, Petri nets present a graphical view of the dynamics of the system.

Systematic techniques developed for Petri net models can be used to study the qualitative aspects of a manufacturing system: absence/presence of deadlock, reinitializability, and buffer overflows [Alla et al. 1985; Kamath and Viswanadham 1986; Likic and Zizkovic 1989; Narahari and Viswanadham 1984]. The introduction of timed transitions has transformed Petri nets into a powerful performance evaluation tool

[Holliday and Vernon 1987; Molloy 1982]. The literature is abundant in methods that use timed Petri net models together with simulation and Markov process models for performance evaluation [Balbo, Chiola, and Franceschinis 1989; Kamath and Viswanadham 1986].

Summary

The literature related to hybrid metamodeling of manufacturing systems has been reviewed. The review was conducted by examining significant works concerning metamodeling of hierarchical systems, hybrid modeling, and performance modeling of manufacturing systems. The results in these areas are impressive.

Metamodeling, as defined in this effort, has been discussed in the literature for almost two decades under various names. Table IV on the next page presents a capsule review of the literature in this area and the most significant distinctions between each work and this effort.

Hybrid modeling or more generally a hybrid approach to modeling is a combination of simulation with analytical modeling. Table V on the next page overviews the hybrid modeling literature considered in this study.

The literature on performance modeling is voluminous. It was reviewed above in four specific areas: discrete event simulation, queueing network models, fast simulation, and Petri nets. Due to the volume of works cited, it is not practical to review this literature in table form. In terms of distinction from the research presented in this dissertation however, all works share a common bond. While most of the works deal with manufacturing issues, none present an approach that unifies hybrid modeling and metamodeling. Fast simulation appears to be the most fertile area in this regard, but to date published results are not available.

TABLE IV
METAMODELING LITERATURE OVERVIEW

AUTHORS	MAJOR THRUST AREA	SIGNIFICANT DIFFERENCES FROM THIS RESEARCH
Meisel and Collins [1973]	Repro-Modeling using composed functions and piecewise linear functions	Not manufacturing specific
Blanning [1975]	Post-simulation sensitivity modeling	No hybrid approach No methodology
Kleijnen [1979] Friedman [1984; 1986]	Regression based methodology	No hybrid approach
Schriber [1987]	Regression analysis	No methodology
Zeigler [1984; 1990] Sevinc [1988]	DEVS scheme aggregation and simplification	Not manufacturing specific Emphasis on form over function

TABLE V
HYBRID MODELING LITERATURE OVERVIEW

AUTHORS	MAJOR THRUST AREA	SIGNIFICANT DIFFERENCES FROM THIS RESEARCH
Schwetman [1977; 1978]	Hybrid modeling using a two phase approach	Requires decomposition Not Manufacturing specific
Shanthikumar and Sargent [1983]	Unified Hybrid modeling class definitions	Not a methodology
Tolopka and Schwetman [1979] Dietrich and March [1985] Nymon [1987] Haider, Noller, and Robey [1986]	Manufacturing examples of hybrid modeling	Decomposition required in each case

This literature review reveals that little controlled experimentation has been documented which focuses on the behavioral performance of hybrid metamodels of hierarchical manufacturing systems. Therefore, this area, as define in Chapters I and II, appears to be a fruitful area for academic investigation.

CHAPTER IV

STATEMENT OF RESEARCH

Research Goal

The principal goal of this research is to develop a methodology for hybrid metamodeling of hierarchical manufacturing systems within a simulation framework. The metamodeling methodology will comprise (1) a procedure to determine which workcenters within a multi-workcenter manufacturing system are candidates for a metamodel, (2) a procedure to create and validate an observation-based metamodel, (3) a procedure to create a queueing network metamodel, and (4) a procedure to implement a metamodel within a manufacturing simulation.

Research Objectives

To accomplish the goal, the following research objectives have been identified:

OBJECTIVE 1 - Metamodel Selection Procedure

Develop a procedure to determine which workcenters within a multi-workcenter simulation model of a hierarchical manufacturing system are candidates for a metamodel. The metamodel decision will be based on (1) the experimental frame being investigated and (2) the availability of a valid plug-compatible metamodel. The metamodel availability question will be based on (1) the workcenter structure (i.e., physical layout and routings), (2) the input to the workcenter (i.e., ratio of arrival rate to service rate), and (3) the workcenter operating characteristics (i.e., alternate routings, breakdowns, multiple

concurrent resource requirements, etc.). The metamodels considered in this research will be those that can be classified as either queueing network metamodels or observation-based metamodels.

OBJECTIVE 2 - Observation-Based Metamodel Procedure

Develop and test a procedure to create and validate (at the workcenter level) a steady-state observation-based metamodel for a workcenter that is amenable to this class of metamodels. The resulting metamodel will be a "plug compatible" replacement for the candidate workcenter within a manufacturing simulation model. The workcenters in this class will fall outside the realm of queueing networks with product form solutions.

Validation of the metamodel at the workcenter level will be accomplished by comparing the base-model and metamodel time-in-system distribution curves. This process will involve a statistical comparison of the two cumulative distribution function (CDF) curves using confidence limits calculated for the base-model curve.

OBJECTIVE 3 - Queueing Network Metamodel Procedure

Develop and test a procedure to create a steady-state queueing network metamodel for a workcenter that is amenable to this class of metamodels. The queueing network models considered within the context of this research will be those with a product form solution and overtake-free paths. The resulting metamodel will be a "plug compatible" replacement for the candidate workcenter within a manufacturing simulation model.

OBJECTIVE 4 - Proof of Concept Via Prototype Implementation

Implement a prototype version of the results of objectives 1 through 3 in a way that demonstrates proof of concept. This implementation will be accomplished using the OSU OOM environment. The procedure developed in Objective 1 will be used to

perform the metamodel applicability and selection analysis. The metamodel development procedures of Objectives 2 and 3 will be used to develop the library of available plug compatible metamodels.

Implementation also will require the development of a procedure for on-line parameterization of the metamodel. The parameter associated with the metamodel during this research is the ratio arrival rate to service rate. The structure and operating characteristics of the metamodel are assumed to be known and constant for a given simulation run. However, the ratio may be dynamic or unknown or both. Through the procedure developed in this step, the metamodel will monitor the value of the ratio and adjust its own operation accordingly during the simulation run.¹

OBJECTIVE 5 - Plant Level Validation

Evaluate the implementation of the methodology and metamodels at the plant level by conducting a series of simulation experiments on a multi-workcenter plant model. One workcenter within the plant model will be compatible with the prototype metamodels developed in Objective 4. Both a base-model version and a metamodel version of the plant model will be evaluated under several decision alternatives. The metamodel version will be judged valid if the decision outcomes resulting from its use are consistent with the decision outcomes resulting from the base-model. The following paragraph illustrates this approach.

Assume that simulation is being used to analyze the impact on time-in-system of speeding-up a processing step for a work flow item (i.e., a part). A comparison is to be made between a 10% speed-up and a 20% speed-up to see if a significant difference exists. Using Figure 7 on the next page as a frame of reference, if the base model is used for this analysis, then the decision can be based on whether the distance "d1" is judged to

¹The need for this procedure eventually became moot due to the particular workcenter configurations used in this study and the steady state assumption for metamodels. This circumstance is discussed more fully in Chapter IX.

be significant or not. Alternatively, if the metamodel version is used, then "d2" must be judged. The validity of the metamodel version can be determined by comparing the two decisions. If d1 is significant then d2 also should be significant. Conversely, if d1 is not significant then neither should d2 be significant. If the metamodel version shows consistency across a range of decision alternatives then it can be judged valid within this context.

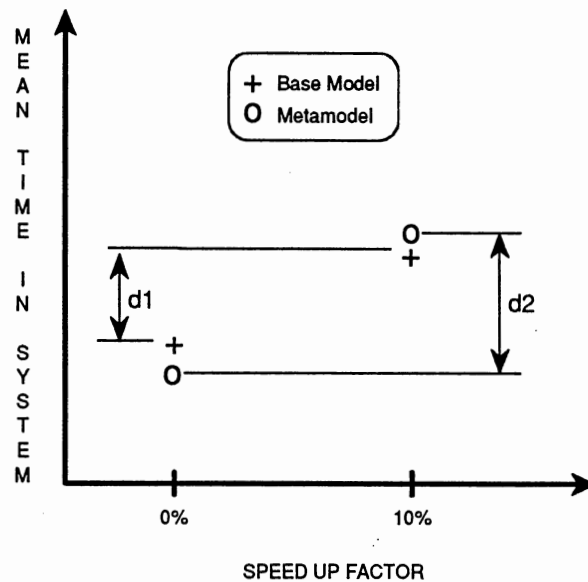


Figure 7. Decision Based Validation

OBJECTIVE 6 - Future Research

Conceptualize a framework for conducting additional research to expand the functionality of the prototype implementation used to demonstrate proof of concept. At

the conclusion of the current effort, much work will yet be required to generalize the methodology and prototype implementation in a more robust environment. It is anticipated that knowledge gained in this effort can be used as a foundation for further research and to provide meaningful guidance as to the most profitable directions.

Research Assumptions

The primary assumption of this research is that bottom-up hierarchical modeling of complex manufacturing systems is both viable and, in many situations, superior to traditional top-down modeling approaches. (Perhaps defining the preferability boundaries of this issue is a legitimate research question by itself.) If one adheres to the traditional top-down single purpose approach to simulation then metamodels are essentially irrelevant. Since the model would be developed to service a single experimental effort, modification of the abstraction level (hence metamodeling) would never arise as an issue.

An additional assumption of this research is that the metamodel question will be considered only on a workcenter by workcenter basis. This is not intended to imply that metamodels could not be created for groups of workcenters (i.e., departments) or that second-order metamodels (i.e., a metamodel of a group of metamodels) should not be investigated. On the contrary, it is anticipated that knowledge gained here can be expanded through additional research to have broader application in the area of simulation modeling of complex manufacturing systems.

Many potential measures of performance can be considered for a manufacturing simulation. This research assumes that the performance measure of primary interest is the time-in-system for work flow items. Further, it is assumed that the distribution of this statistic is required. In other words, besides estimates of the mean and variance of time-in-system, the modeler is interested in percentile measures of its dispersion. As was

stated above, it is this assumption that forces potential queueing network metamodels to have a product form solution.

Several different workcenter variables could justifiably be considered for parameterizing the metamodels. Among the most logical candidates, based on queueing literature, are mean arrival rate, arrival rate variability, mean service rate, and service rate variability. For this research, the author assumes that the single most significant variable in a workcenter model is the ratio of mean arrival rate to mean service rate. This assumption is based on the central role that this ratio (typically symbolized by ρ) plays in product form solutions of queueing networks. The mean service rate for a multi-station workcenter with series stages will be the service rate of the slowest stage. The slowest stage will be used since it represents the bottleneck stage and generally the controlling rate for the workcenter throughput.

Since the primary goal of this research is the development of a methodology, the experimental design for validation is directed toward proof of concept rather than toward the development of a comprehensive library of metamodels. The test cases presented represent a realistic set of scenarios that have practical merit based upon the author's experience in industry.

Research Contributions

The major contribution anticipated from this research is the conceptualization and validation of a methodology to create observation-based metamodels for use in hybrid simulation of complex manufacturing systems. For modeling practitioners, the development of this methodology offers significant rewards in two areas. First, enhancement of the computational efficiency of simulation facilitates its use in on-line real-time decision making. This area offers potentially large rewards and yet remains virtually untapped by simulation due to lengthy execution times. Second, the development of metamodels will hopefully lead to the creation of basic knowledge about systems

and operations. This basic knowledge is in the form of insights regarding the functional form of the relationships between input and performance variables of a workcenter. It is this sentiment that is reflected in the statements of Hamming [1962], Geoffrion [1976], and Ignall and Kolesar [1978] who each paraphrased the maxim that "the purpose of modeling is insight, not numbers."

Other contributions anticipated from this research include:

- o Demonstration of the viability of plug-compatible alternate representations of coupled models within the OSU OOM environment.
- o Demonstration of the viability of hybrid modeling within the OSU OOM environment.
- o Conceptualization and implementation of a procedure to initialize a parameterized metamodel although the steady-state value of the parameter is unknown and/or dynamic.
- o Demonstration of the OSU OOM environment as a research test bed.

CHAPTER V

RESEARCH METHODOLOGY

Performance Measure

Within the scope of this research, the performance measure of primary interest is the time-in-system for work flow items (parts). In related literature this performance measure is sometimes designated as the sojourn time or the passage time. This research addresses the statistical distribution of the time-in-system random variable, not just the mean and the variance as is frequently the case. The research experiments described in the following section collect data from which an empirical cumulative distribution function of time-in-system is constructed. This empirical function becomes the basis of the metamodel validation procedure.

Research Plan

To achieve the goals and objectives outlined in Chapter IV, the research will be performed in phases as detailed below. There are eight major phases. Phase I finalizes the experimental factors that will be used in the subsequent development and validation phases. Phase II develops and validates the observation-based metamodel procedures. It is anticipated that this phase will be the most intellectually challenging. Phase III develops the queueing network metamodels. This phase will build from known, published solutions and is included to demonstrate the robustness of the hybrid metamodeling approach. Phase IV develops the rule based metamodel selection procedure. This phase is perhaps the second most intellectually challenging. Phase V is

the prototype implementation. The most significant challenge of this phase will be conquering the subtleties of object oriented programming to bring the intellectual accomplishments to fruition. Phase VI validates the prototype hybrid metamodeling methodology. This phase is the longest phase. It is primarily devoted to running simulation models and analyzing the decision-based results. During this phase the merits of the implemented prototype of the methodology will be evaluated. Phases VII and VIII are the wrap-up phases. Phase VII is devoted to identifying areas of future research. Phase VIII represents the culmination of the research through preparation of the final dissertation document. The phase dependency relationships are presented in a subsequent section. Detailed phase and task descriptions follow immediately.

PHASE I - Finalize Experimental Factors

The goal of this phase is to finalize the specification of experimental factors that will be used during metamodel development and plant level validation. The factors (and anticipated number of different levels) are outlined below. Additional detail regarding the levels of the factors is presented in the sections titled: Observation-Based Metamodels, Queueing Network Metamodels, and Plant Level Decision-Based Validation.

- o Factors related to observation-based metamodel development:
 - workcenter structure; number of machines and routings; 1 value
 - workcenter input values; arrival rate to service rate ratios; 4 values
 - sets of workcenter operating characteristics; 4 values
 - run repetitions to generate time-in-system statistics; 5 values
- o Factors related to observation-based metamodel validation:
 - workcenter input values (arrival rate to service rate ratios) ; 2 values
 - run repetitions to validate metamodels; 5 values
- o Factors related to plant level metamodel validation:
 - plant structure; structure of non-metamodel workcenters; 1 value

decision alternatives; Case I - 2 alternatives; Case II - 2 alternatives
plant input values (arrival rates) 2 values.

The combination of factors described above will result in a minimum of 480 simulation runs (breakdown provided below) to complete the experimental portion of the research. Running on an IBM PS/2 Model 70 personal computer, each workcenter level validation run is expected to take approximately 15 minutes; each plant level run 30 minutes. This estimate brings the total simulation time to 210 hours.¹

PHASE II - Observation-Based Metamodels

Task 1: Construct the base workcenter models to be used in developing the observation-based metamodels. There will be one base workcenter model (and subsequently one observation-based metamodel) for each combination of workcenter structure X operating characteristics (4 combinations).

Task 2: Run the base workcenter models to collect the time-in-system statistics. Eighty simulation runs will be required (4 combinations X 4 input values X 5 reps).

Task 3: Analyze the time-in-system statistics to develop the four observation-based metamodels. The section title "Observation-Based Metamodel Development Procedure" later in this chapter outlines this procedure.

Task 4: Validate the observation-based metamodels. Forty new simulation runs will be required to generate time-in-system statistics for two new input values (4 combinations X 2 input values X 5 reps). The eight simulated time-in-system curves (4 combinations X 2 input values) will be statistically compared to the eight corresponding curves generated by the metamodels.

¹The estimated run times proved quite accurate. Unfortunately, the number of runs required, and the resulting total execution time, increased significantly due to the nature of the validation procedure discussed in Chapter VII. The availability of multiple computers on which to simultaneously execute simulation runs served as a mitigating factor.

PHASE III - Queueing Network Metamodels

Task 1: Construct the base workcenter models to be used in developing the queueing network metamodels. There will be one base workcenter model (and subsequently one queueing network metamodel) for each of the two workcenter configurations.

Task 2: Formulate the two queueing network base models as queueing networks with a product form solution, then solve the networks to develop the metamodels. The section title "Queueing Network Metamodel Development Procedure" later in this chapter outlines this procedure. No validation step is included since the metamodels are based on previously published results from the queueing literature.²

PHASE IV - Metamodel Selection Procedure

Develop the procedure to determine which workcenters within a multi-workcenter manufacturing system are candidates for a metamodel. This rule-based decision procedure will consider the experimental frame being investigated and the availability of a valid plug-compatible metamodel.

PHASE V - Prototype Implementation

Task 1: Develop the procedure for on-line initialization of the metamodel. A subsequent section titled "Metamodel Initialization Procedure" outlines the development of this procedure.³

Task 2: Implement a prototype metamodeling capability within the OSU OOM environment to demonstrate proof of concept. This phase will be primarily

²In reality the "Simulation Run Design Considerations" in Appendix B are nothing more than a validation procedure for the queueing network metamodels.

³As stated in Chapter IV, the need for this procedure eventually became moot due to the particular workcenter configurations used in this study and the steady state assumption for metamodels.

devoted to writing and/or modifying the Smalltalk-80 code required to implement the procedures and results above.

PHASE VI - Prototype Validation

Task 1: Construct the base plant model to be used in validating the metamodels at the plant level. A subsequent section titled "Plant Level Prototype Validation" outlines this procedure. There will be six basic plant models. Each will contain three workcenters. Workcenters 1 and 3 will be unchanged in each of the four plant models. Workcenter 2 will be a workcenter corresponding to the structure and operating characteristics of each of the base workcenter models used in Phases II and III during the development of the metamodels (6 variations).

Task 2: Run the plant models to collect mean time-in-system statistics for each decision alternative. One hundred and eighty simulation runs will be required (3 decision alternatives⁴ X 6 variations X 2 input values X 5 reps).

Task 3: Repeat tasks 1 and 2 using the metamodels developed during Phases II and III. One hundred and eighty additional simulation runs will be required (3 decision alternatives X 6 metamodels X 2 input values X 5 reps).

Task 4: Validate the metamodeling methodology by comparing the decision outcomes from the alternative comparisons. The comparisons will be made in the following pairwise manner (the Cases and Alternatives are defined in a subsequent section titled "Plant Level Prototype Validation"):

Decision 1: Case I-1 versus Case I-2

Decision 2: Case II-1 versus Case II-2.

For each decision the mean time-in-system from the base model runs for each of the two alternatives will be compared to decide if a significant difference exists. The same

⁴Case I - Alternative 2 and Case II - Alternative 1 are actually the same alternative thus there are only three unique alternatives.

comparison will be made for the mean time-in-system from the metamodel runs. For the metamodels to be judged valid, the significant/not-significant decisions must be consistent.

PHASE VII - Framework for Future Research

Develop a long term framework providing direction for future research in this area. At the conclusion of the previous phases, a prototype implementation of the methodology will have been achieved and its validity tested. To gain full benefit from this methodology and provide additional (more robust) functionality, a planned approach to additional research is required. This phase will outline a coherent, consistent approach in this regard.

PHASE VIII - Summarize Results and Prepare Final Format

Summarize and document the research results. This phase represents the culmination of the research activities and the presentation of results in final form.

Observation-Based Metamodel Scenarios

The purpose of this section is to document the workcenter structure, operating characteristics and workcenter inputs that are to be evaluated during the observation-based metamodel development phase.

As stated earlier, observation-based metamodels are utilized when a workcenter cannot be formulated as a queueing network with a product form solution. The following characteristics are among those that violate the product form solution assumptions:

- o load-dependent alternate routings (i.e., based on workcenter queue lengths)
- o multiple concurrent resource requirements (i.e., machine and operator, etc.)
- o machine breakdowns

- o finite buffers leading to blocking of machines.

The observation-based metamodels for this research will be created by taking a single workcenter structure (number of machines and primary routings) and adding each of the above operating characteristics in turn to create four non-product form scenarios. The workcenter structure to be used throughout will be a two stage structure composed of a single machine followed by three parallel alternates as illustrated in Figure 8 on the next page. Each work flow item entering the workcenter requires an operation on the single machine followed by an operation from one of the three parallel alternates. The stages will be balanced in terms of expected throughput.

Four different levels of workcenter input will be used to develop the parameterized observation-based metamodels. The input variable will be the ratio of workcenter arrival rate to stage service rate, ρ . The four levels to be used are: 0.25, 0.40, 0.60, and 0.80. To validate the metamodels, input values of 0.50 and 0.75 will be used.

The observation-based metamodel scenarios can be summarized as follows:

Characteristics Common to All Scenarios

ρ - Arrival Rate/Service Rate Ratio (Development): 0.25, 0.40, 0.60, 0.80;

ρ - Arrival Rate/Service Rate Ratio (Validation): 0.50, 0.75;

Arrival Process: Poisson with rate based on ρ ;

Service Rate Distributions: Triangular; stage mean: 1 time unit: minimum and maximum at mean $\pm 10\%$;

Workcenters are dedicated to a single work flow item (i.e., no part mix);

Lot sizes are fixed at 1;

Material Handling time is fixed at zero;

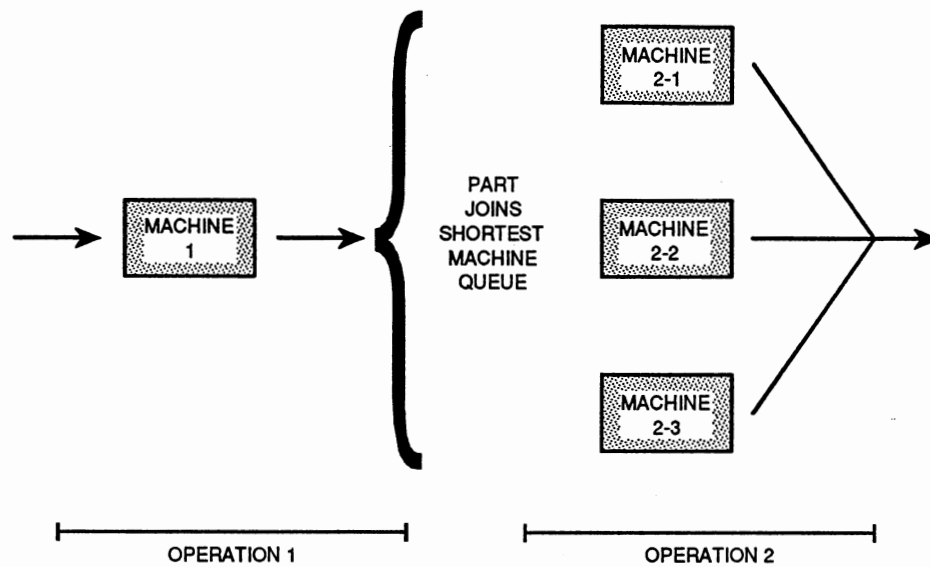


Figure 8. Observation-Based Metamodel Workcenter Structure

Scenario 1 (State Dependent Alternate Routing)

For operation 2, the work flow item is routed to the parallel alternate with the shortest waiting queue.

Scenario 2 (Scenario 1 plus Multiple Concurrent Resources)

For all operations, the machines require an operator assisted set-up. Set-up time is deterministic and 10% of the mean service time for each stage. A single operator services the entire workcenter.

Scenario 3 (Scenario 2 plus Machine Breakdowns)

All machines are subject to breakdown. Breakdowns require the operator full time until the repair is completed. Breakdown and repair distributions are Triangular with parameters to be set based on exploratory runs.

Scenario 4 (Scenario 3 plus Finite Queues)

The three parallel alternate machines required for stage 2 are all preceded by finite queues (queue capacity to be determined in exploratory runs). If all alternate machine queues are full, machine 1 becomes blocked.

Queueing Network Metamodel Scenarios

The purpose of this section is to document the workcenter structure, operating characteristics and workcenter inputs that are to be evaluated during the queueing network metamodel development phase.

These scenarios involve workcenter structures that are amenable to queueing network analysis with product form solutions. Further, these scenarios will involve workcenters with overtake-free paths so that known results from the literature can be used to calculate the time-in-system distributions.

Characteristics Common to Both Scenarios

ρ - Arrival Rate/Service Rate Ratio (Development): 0.25, 0.40, 0.60, 0.80;

Arrival Process: Poisson with rate based on ρ ;

Service Rate Distributions: Exponential; stage mean: 1 time unit;

Workcenters are dedicated to a single work flow item (i.e., no part mix);

Lot sizes are fixed at 1;

Material handling time is fixed at zero;

Scenario 1 (Tandem Network)

This tandem network will comprise three machines in series. Work flow items must visit each machine in sequence before exiting the workcenter.

Scenario 2 (Tree Network)

This tree network will comprise one machine followed by two machines with probabilistic routing. The probabilistic routing will be random (i.e., a 50/50 chance of being routed to either stage 2 machine).

Plant Level Prototype Validation

The purpose of this section is to document the plant structure and decision alternatives that are to be evaluated during the plant level validation phase. The basic plant model will contain three workcenters as shown in Figure 9 below. Workcenters 1 and 3 will remain unchanged in each plant validation run. Workcenter 2 will be a workcenter corresponding to the structure and operating characteristics of each of the base workcenter models used in Phases II and III during the development of the metamodels. The exact configuration of workcenters 1 and 3 is illustrated in Figure 10 on the next page.

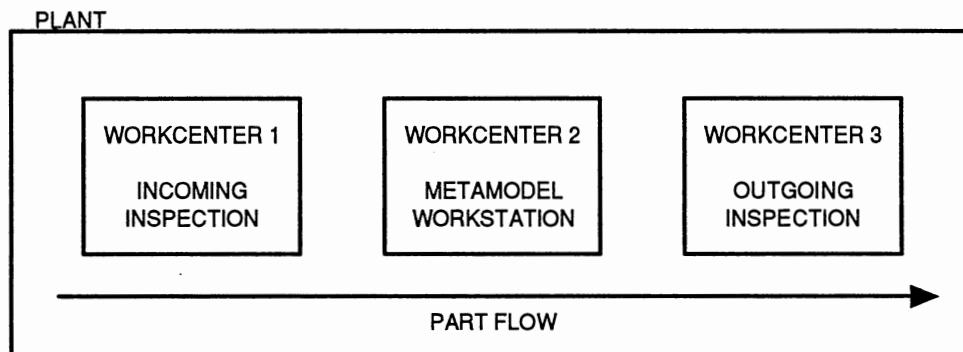


Figure 9. Basic Configuration Of Plant Model

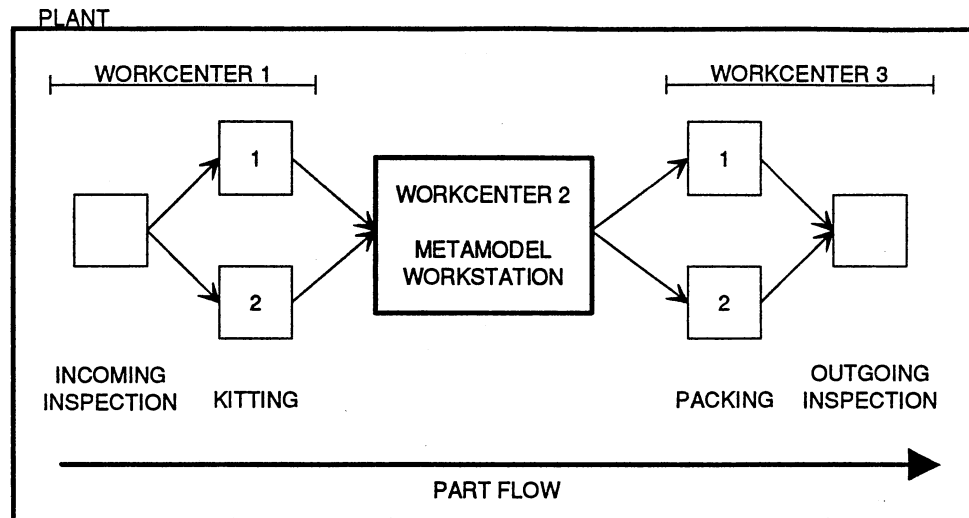


Figure 10. Detailed Configuration Of Plant Model

Two decision cases will be evaluated comprising four decision alternatives. Case I decisions concern the merits of speeding up the incoming and final inspection stations in workcenters 1 and 3. The two alternatives to be evaluated are: leave the inspection stations as-is (a.k.a., slow) versus speed-up the inspection stations by reducing the mean service time by 10% (a.k.a., fast). Case II decisions concern the queue discipline strategy used in workcenters 1 and 3. The two alternatives to be evaluated are: always route parts to the shortest available queue (a.k.a., shortest) versus preferential queuing to machine 2 unless its waiting queue is three or more longer than machine 1 (a.k.a., preferred).

The plant level decision-based prototype validation can be summarized as follows:

Case I-1: No Inspection Station Speed-Up

Case I-2: Speed-Up Inspection Station

Case II-1: Workstations 1&3 Queue Strategy: Shortest Available

Case II-2: Workstations 1&3 Queue Strategy: Preferential

Observation-Based Metamodel Development Procedures

The purpose of this section is to document a preliminary outline for the observation-based metamodel development procedure. The metamodels for the observation-based class will be developed using the following general approach. For each combination of workcenter structure and operating characteristics (see Observation-Based Metamodel Scenarios above), do the following:

Step 1 - Base Model Data Generation

For a given workcenter input value (ρ), run five repetitions (reps) of the base model workcenter structure and collect time-in-system observations.

Step 2 - Plot Data Points

For each rep, plot percentile points for the time-in-system statistic. The x-axis of this plot will represent time-in-system (TIS) values. The y-axis will represent cumulative percentage of the collected time-in-system values (i.e., a CDF axis expressed in percent). The plot points along the x-axis will be at cells of a fixed width. Developing the plot in this fashion will result in 5 y-values (one per rep) for each x-value (cell). Figure 11 on the next page illustrates the plot generated by this step of the procedure (including the average curve to be generated by the next step).

Step 3 - Regression

Average the five CDF values in each cell and plot a single average CDF curve for TIS.

Step 4 - Repeat For All Input Values

Repeat steps 1 through 3 for all settings of the workcenter input value, ρ .

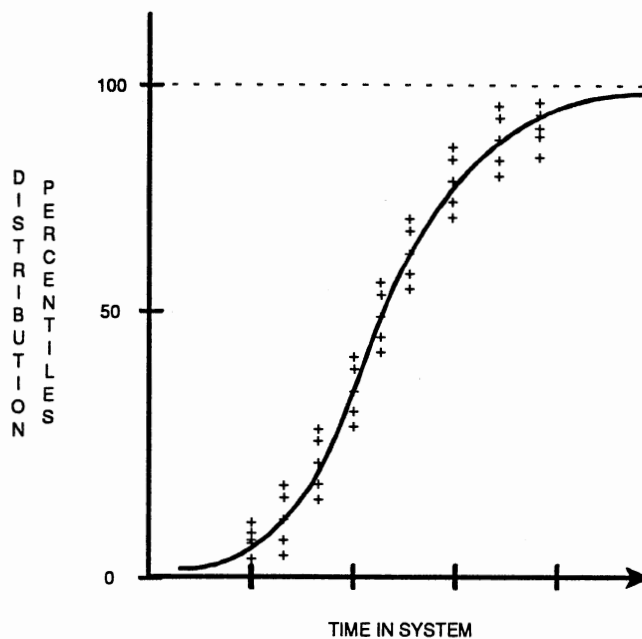


Figure 11. Plot Of Observed Time-In-System Distribution

Step 5.- Group CDF Curves

Plot all curves generated by repetitions of Step 3 on a common graph.

Figure 12 on the next page illustrates an example of the family of curves developed within this step.

Step 6.- Develop Metamodel

Attempt to parameterize the CDF curves. Ideally this process will yield a parameterized function of TIS in terms of CDF where the parameters are determined by the input values. This parameterized function becomes the metamodel from which time-in-workcenter samples can be drawn via the inverse transformation method. If the curves are not amenable to parameterization, the set of curves will be used to interpolate a value for time-in-workcenter samples via inverse transformation.

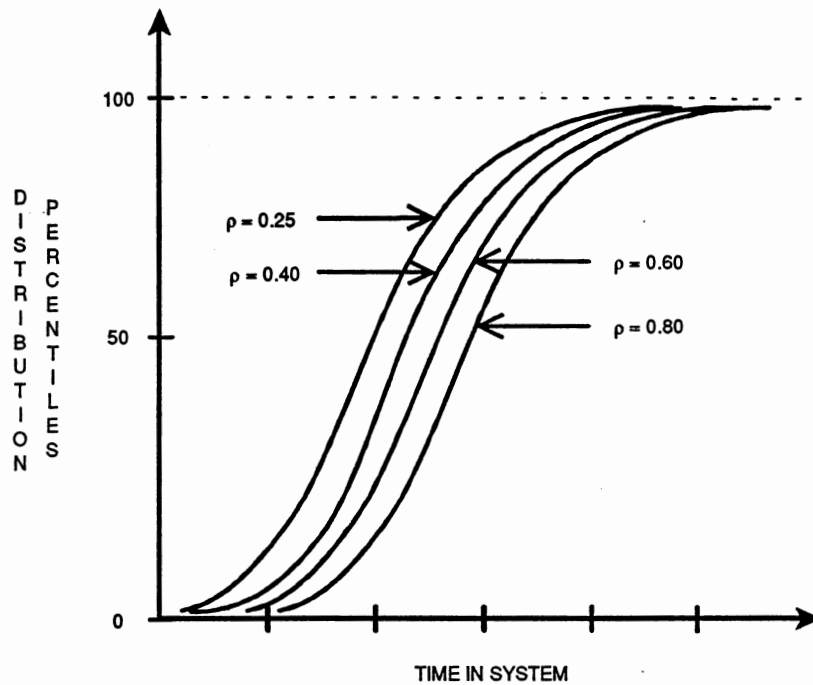


Figure 12. CDF Curves For A Workcenter Group

Queueing Network Metamodel Development Procedure

The metamodels for the queueing-network class will be developed using the following general approach.

Step 1 - Queueing Network Formulation

Formulate the workcenter model as a product form queueing network.

Step 2 - Solve For The Time-In-System Distribution

Using the methodologies and algorithms developed by Walrand and Varaiya [1980], Daduna [1982], Boxma, Kelly, and Konheim [1984], and Kelly and Pollett [1983] solve for the time-in-system distributions.

Step 3 - Calculate The Time-In-System CDF Curve

Using the time-in-system distribution, calculate the CDF function.

Step 4 - Develop Metamodel

The CDF function becomes the metamodel from which time-in-system samples can be drawn via the inverse transformation method.

Metamodel Initialization Procedure⁵

Within the context of this research, a metamodel is a function that is parameterized on the ratio of arrival rate to service rate. The service rate is internal to the metamodeled workcenter and therefore must be constant within the experimental frame or else a metamodel would not be appropriate. As a result, the metamodel is parameterized on the ratio of arrival rate (expressed as a ratio to a constant value). Occasions may arise in which a metamodel is used but no initial estimate of the arrival rate is available (e.g., changes are made "upstream" and their impact on arrival rate is not known). In these cases a procedure is needed to permit the metamodel to self-adapt to the ratio it is realizing during the simulation. This is essentially a warm-up procedure that allows the metamodel to approach steady state as the rest of the model (and resulting arrival rate at the metamodel) approaches steady state.

Two basic options for metamodel parameterization will be investigated in a cursory fashion. If the models prove sensitive to this initialization procedure, additional research will be proposed within Objective 6. The two basic options to be considered are:

Option 1: Using exponential smoothing, maintain a weighted estimate of the current arrival rate based on the realized rates since the start of the simulation run. If no initial estimate is available for the "time zero" estimate, a value of 0.5 will be used. A reasonable value for the smoothing constant will be determined during development.

⁵As stated in Chapter IV, the need for this procedure eventually became moot due to the particular workcenter configurations used in this study and the steady state assumption for metamodels. This circumstance is discussed more fully in Chapter IX.

Option 2: Modify option 1 such that when the arrival rate estimate reaches steady-state the current estimate will henceforth be maintained as a global average of all values since achieving steady-state, rather than as an exponentially smoothed average. The method of detecting when the arrival rate has achieved steady state will be determined during development.

CHAPTER VI

OBJECT ORIENTED REPRESENTATION

Introduction

This chapter presents an overview of the object oriented environment that was used to implement and test the methodology proposed in Chapter V. The general characteristics and desirable features of an object oriented modeling (OOM) environment were presented in Chapter II. The OOM advanced modeling environment upon which this research is based has been evolving at OSU's Center for Computer Integrated Manufacturing (CIM) for approximately five years. Several previous studies have demonstrated its usefulness in analyzing manufacturing systems such as the ones used in this research [Beaumariage 1990; Karacal 1990; Basnet 1991].

Object Oriented Classes

The OSU OOM environment operates under Objectworks For DOS (Version 4.0), an implementation of Smalltalk-80 [Goldberg 1989] designed specifically to run on personal computers running the DOS operating system. Smalltalk-80 is one of the purest object oriented languages in that it adheres rather strictly to the object paradigm discussed in Chapter II. The initial classes underlying the OSU OOM advanced modeling environment were developed using an early version of Smalltalk-80 for the PC (Version 2.5). Significant enhancements have been made to these original classes and methods as a result of on-going research within the Center for CIM. A comprehensive review of the OSU advanced modeling environment is beyond the scope and need of this

effort. The interested reader is referred to Basnet et al. [Library of Objects, 1990] for additional detail. Following is a brief overview of the primary classes needed for the metamodeling evaluation. Figure 13 on the next page summarizes these classes in a hierarchy tree. The tree highlights the inheritance relationships between the classes and in some case shows relevant instance variables in parenthesis following the class name.

- o SimModel. This class provides the overall framework for the simulation model including managing the user interface and launching the simulation process. It is the driver of the "model" component of the Smalltalk-80 model-view-controller (MVC) trilogy [Goldberg 1989] for the advanced modeling environment.
- o SimView. This class provides the primary user interface, the "Simulation Launcher". It is the driver of the "view" component of the Smalltalk-80 MVC trilogy.
- o CimSimulation. This class is a subclass of the Smalltalk-80 supplied Simulation class. It internally manages the simulation process through actions such as manipulating the event queue and pausing or resuming processes.
- o Plant, WorkCenter, WorkStation. This hierarchy of classes provides for the physical resources required to process parts. They also facilitate the natural flow of decision and control up and down the hierarchy as required. An important subclass of WorkStation is Operator.
- o WorkflowItem. Work flow items represent the parts which require processing by the plant. A work flow item accesses the routing dictionary to find the required operations for its part name and visits the required workstations. Work flow items are represented internally (to the advanced modeling environment) as "processes" that may be suspended and resumed as resource availability dictates.

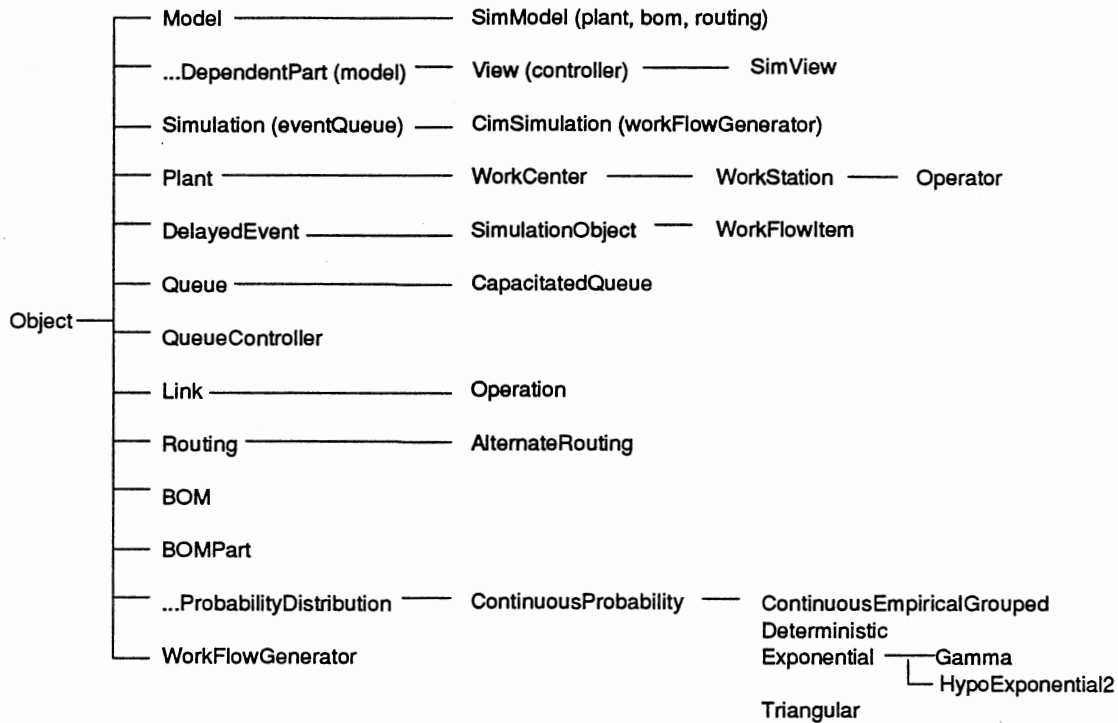


Figure 13. OOM Class Hierarchy Diagram For Metamodeling

- o Queue. This class provides a mechanism to account for items awaiting processing. Queues can be physical or logical. Physical queues, like the input and output queues of a machine, provide WIP storage space as well as the logical sequencing of items. Logical queues, such as the input queue of a material handler or operator provide only logical sequencing. An important subclass, *CapacitatedQueue*, provides queues with finite capacities.
- o QueueController. Queue controllers provide the vehicle through which a workstation communicates with its input and output queues. The controller is also responsible for implementing queue discipline logic.

- o **Operation.** An operation is a data structure that specifies a machine, a processing time, and a setup time required by a work flow item to be processed on that machine.
- o **Routing.** A routing is a collection of operations that specifies the series of steps required to complete the processing of a `WorkFlowItem`. An important subclass of `Routing` is `AlternateRouting` that specifies an operation on a different machine that can be substituted for the primary operation.
- o **BOMPart.** An instance of this class represents a component within a bill of materials hierarchy tree.
- o **BOM.** This class represents the entire bill of materials hierarchy. A list of `BOMParts` is maintained to represent the complete list of parts in the plant.
- o **ProbabilityDistribution, ContinuousProbability.** This hierarchy of classes provides the framework for generation of random variates. The variates themselves are generated by subclasses `Exponential`, `Triangular`, `Deterministic`, `Gamma`, `HypoExponential2`, and `ContinuousEmpiricalGrouped` using the inverse transformation method. Class `Random` is used to generate the uniform random numbers used in inverse transformation.
- o **WorkFlowGenerator.** This class creates work flow items and sends them into the system to be processed.

Changes Made To The Environment

Several changes were required within the advanced modeling environment to facilitate this research. Table VI on the next page summarizes the most significant of these changes. In many cases the changes were made in an application specific manner. To the extent that the changes are of generic interest, with some additional work, they can be generalized for inclusion in the permanent OOM environment. The relevant

TABLE VI
SIGNIFICANT OOM ENVIRONMENT CHANGES

AREA	DESCRIPTION	CLASS MODIFIED
Random Variate Generation	Added new random number generator ¹	Random
	Modified method for variate generation ¹	Gamma
	Added multiple stream variate generation capability	Exponential Gamma
	Added new variate generator ¹	Triangular, HypoExponential2, ContinuousEmpiricalGrouped
	Added Global Variables: TheSeedArray, RandomRouter, MetaRhos,MetaCells,MetaValues	SimModel, WorkStation, WorkCenter, Utils
Queueing Selection Strategy	Added Preferred Queue Strategy	Plant WorkCenter
	Added Random Queue Strategy	WorkCenter
Simulation Termination	Terminate based on number of values collected rather than time	SimModel
Metamodel Implementation	Replace a workcenter with a metamodel	SimModel
	Enable metamodel with infinite availability	WorkStation
	Read and process metamodel file	Utils
	Calculate metamodel sampling distribution	Utils
Operator Implementation	Define the Operator resource	Operator
	Allow a workcenter to possess an operator resource	WorkCenter
	Enable a work flow item to request a second resource	WorkFlowItem CimSimulation
Queue Length Inquiry	Include machine status in queue length query	WorkCenter
Blocking	Allow blocking to consider alternate workstations	WorkCenter

¹Special thanks go to Steve Tretheway and Mike Oltman from The University of Oklahoma for their research and implementation of the random number and random deviate generators.

portions of new and/or modified Smalltalk-80 code (classes and methods) are provided in Appendix F for the interested reader.

Conducting An OOM Experiment

Four basic steps are required to conduct an experiment within the advanced modeling environment; plant definition, BOM definition, routing definition, and experimental parameters definition. User interface options exist to bypass the first three steps by reading from disk a set of previously defined files that contain the plant, BOM, and routing definitions. The definition of experimental parameters is always conducted on an on-line basis.

Defining the plant is accomplished in two phases. First, the names and hierarchical relationships of the physical resources within the plant must be defined. The hierarchical relationships specify (1) the workcenters within the plant and (2) the workstations within the workcenter. Figure 14 on the next page illustrates this structure for one of the plant configurations (scenario OB3) used within this research. The second phase of plant definition is to specify the failure and repair distributions for each workstation defined in the plant. Selecting the distribution and specifying the associated parameters is handled by the user interface via scrolling lists and fill-in-the-blank prompts. The "filed out" version of the complete plant definition for scenario OB3 is shown in Figure 15 on page 77.

The second major step in conducting an experiment is the BOM definition. Bills of materials are specified in two phases. First, a list of part names is created. This list includes all inventoried items at all levels within the plant. The second step is to specify the component breakdown of each part name. The component breakdown is specified through parent/child relationships. A "quantity per" value also can be designated. Any part without a parent is assumed to be an end item. Any part without a child is assumed

to be a purchased part. For this research, the BOM for every scenario consists of a single item unobtrusively named "Part". "Part" has no parents or children, it simply enters the plant, proceeds through a series of operations, and exits the plant. Figure 16 on page 77 shows the "filed out" version of the bill of material definition.

```
Plant: PnlOB3
  WorkCenter: WC1
    WorkStation: Inc_Insp
    WorkStation: Kit1
    WorkStation: Kit2
  WorkCenter: OB1
    WorkStation: M1
    WorkStation: M2
    WorkStation: M3
    WorkStation: M4
  WorkCenter: WC3
    WorkStation: Pack1
    WorkStation: Pack2
    WorkStation: Out_Insp
```

Figure 14. OOM Plant Structure Definition

The third major step in conducting an experiment is to define the routings and alternate routings. This also is a two phase process. First, the sequence of workstations to be visited (and any alternates) is specified by selecting from a scrolling list of the workstations defined in the plant definition step. Table VII (page 78) illustrates this sequence for one of the plant configurations (scenario OB1) used within this research. The second phase of routing definition is to specify the processing and setup time

plntob3l	3	InterfaceWorkStation m4l
no commentl	InterfaceWorkStation	no commentl
no controll	out_inspl	no controll
None	no commentl	0 0
3	no controll	failure distributionl Triangular
wc1l	0 0	300 300 450
no commentl	no failurel	repair distributionl Triangular
no controll	no repairl	1.5 1.5 3.0
None	InterfaceWorkStation pack1l	InterfaceWorkStation m1l
3	no commentl	no commentl
InterfaceWorkStation kit2l	no controll	no controll
no commentl	0 0	0 0
no controll	no failurel	failure distributionl Triangular
0 0	no repairl	100 100 150
no failurel	InterfaceWorkStation pack2l	repair distributionl Triangular
no repairl	no commentl	0.5 0.5 1.0
InterfaceWorkStation	no controll	InterfaceWorkStation m2l
inc_inspl	0 0	no commentl
no commentl	no failurel	no controll
no controll	no repairl	0 0
0 0	5	failure distributionl Triangular
no failurel	out_inspl 0	300 300 450
no repairl	pack1l 0	repair distributionl Triangular
InterfaceWorkStation kit1l	inputl 1	1.5 1.5 3.0
no commentl	outputl 30	6
no controll	outputl 1	outputl 1
0 0	inputl 30	inputl 30
no failurel	pack2l 0	m1l 0
no repairl	ob3l	m2l 0
5	no commentl	inputl 1
inputl 1	no controll	outputl 30
outputl 30	None	m3l 0
kit2l 0	4	m4l 0
inc_inspl 0	InterfaceWorkStation m3l	5
outputl 1	no commentl	outputl 1
inputl 30	no controll	inputl 0
kit1l 0	0 0	wc1l 0
wc3l	failure distributionl Triangular	wc3l 0
no commentl	300 300 450	inputl 1
no controll	repair distributionl Triangular	outputl 0
None	1.5 1.5 3.0	ob3l 0

Figure 15. Scenario OB3 Plant Definition File

1
Partl 0

Figure 16. BOM Definition File For All Scenarios

TABLE VII
OOM ROUTING DEFINITION

PART NAME	PRIMARY WORKSTATION	ALTERNATE WORKSTATION
Part	Incoming-Inspection	
	Kit1	Kit2
	M1	
	M2	M3 M4
	Pack1	Pack2
	Outgoing-Inspection	

distributions for each primary and alternate workstation for each operation. Selecting the distribution and specifying the associated parameters are handled by the user interface via scrolling lists and fill-in-the-blank prompts. The "filed out" version of the complete routing definition for scenario OB3 is shown in Figure 17 below.

```

1 Part1 Exponential 0.5
1
Part1
0 1
Part1 6
1 inc_insp1 processTime| Triangular 0.9 1.0 1.1
setUpTime| Deterministic 0.0
2 kit1| processTime| Triangular 1.8 2.0 2.2
setUpTime| Deterministic 0.0
kit2| processTime| Triangular 1.8 2.0 2.2
setUpTime| Deterministic 0.0
1 m1| processTime| Triangular 0.8 0.9 1.0
setUpTime| Deterministic 0.1
3 m2| processTime| Triangular 2.4 2.7 3.0
setUpTime| Deterministic 0.3
m3| processTime| Triangular 2.4 2.7 3.0
setUpTime| Deterministic 0.3
m4| processTime| Triangular 2.4 2.7 3.0
setUpTime| Deterministic 0.3
2 pack1| processTime| Triangular 1.8 2.0 2.2
setUpTime| Deterministic 0.0
pack2| processTime| Triangular 1.8 2.0 2.2
setUpTime| Deterministic 0.0
1 out_insp1 processTime| Triangular 0.9 1.0 1.1
setUpTime| Deterministic 0.0

```

Figure 17. Scenario OB3 Routing Definition File

The fourth step in conducting an experiment is the definition of the experimental parameters. Four parameters must be specified for each run within the context of this research. First, the random number set to be used in initializing the stochastic processes must be specified (five sets are available). Second, length of the simulation run must be specified. The length is specified in terms of the number of time-in-system values collected for "Part". Next, the simulation warm-up period is specified. The warm-up period is specified in number of time units. When the simulation clock reaches the end of the warm-up period, all statistical arrays are cleared. Finally, the variables for which detailed observation data is to be collected must be specified. The specification is made from a scrolling list of available variables. For this research, the sole variable of interest is "Part" time-in-system.

At the conclusion of each experiment, two actions were taken. First, a time series plot of the collected time-in-system values was examined to ensure (visually) that any warm-up effects had been successfully eliminated. Second, output files were written containing the summary statistics for the run and the detailed observations which were collected.

CHAPTER VII

METAMODEL SELECTION PROCEDURE

Introduction

This chapter presents the metamodel selection procedure in partial fulfillment of research objective 1 presented in Chapter IV. As stated previously, the metamodel selection procedure is based on (1) the experimental frame being investigated and (2) the availability of a valid plug-compatible metamodel. The metamodel availability question is based on (1) the workcenter structure (i.e., physical layout and routings), (2) the input to the workcenter (i.e., ratio of arrival rate to service rate), and (3) the workcenter operating characteristics (i.e., alternate routings, breakdowns, multiple concurrent resource requirements, etc.).

The procedure is presented below as a sequence of steps that result in either the recommendation of a suggested metamodel or the decision that metamodeling is inappropriate for that particular workcenter. The procedure has two distinct phases. The first phase determines which workcenters (if any) are amenable to metamodeling based on the experimental frame and the capabilities of the advanced modeling environment discussed in Chapter VI. This phase is performed external to the advanced modeling environment. The second phase determines the availability of a plug-compatible metamodel. This phase is performed internally within the advanced modeling environment.

Assessment of Candidate Workcenters

The first phase determines which workcenters (if any) are amenable to meta-modeling within the framework of the OOM advanced modeling environment. The basis of this determination is the experimental frame. In this context, the experimental frame can be thought of as a statement from the user as to what measures of performance are required as output from the simulation analysis. Law [1986] has identified a list of ten measures of performance that are most often used in manufacturing simulation studies. Table VIII lists these ten measures.

TABLE VIII
MANUFACTURING MEASURES OF PERFORMANCE

PERFORMANCE MEASURE
Part/Job Throughput
Time-In-System for Parts/Jobs
Time-In-Queue for Parts/Jobs
Time-In-Transport for Parts/Jobs
Sizes of WIP Inventory
Utilization of Equipment/Personnel
Proportion of Time that a Machine is Broken, Blocked, or Starved
Proportion of Jobs Produced that must be Reworked or Scrapped
Return On Investment of a New or Modified Manufacturing System
Payback Period of a New or Modified Manufacturing System

In light of the above, two issues must be considered in order to resolve the metamodel amenability question. First, do the experimental measures of performance sought conflict with the aggregation implicit in a given metamodel of a workcenter?

Second, does the OOM environment presented in Chapter VI support investigation of the performance measure?

As stated in Chapter IV, a self imposed restriction of this research is that the measure of performance to be approximately maintained by a metamodel of a workcenter is the distribution of time-in-system¹. In general, this implies that all other detailed behaviors within the workcenter are lost in aggregation. In terms of the ten performance measures, this aggregate performance would eliminate a workcenter from metamodel consideration if one or more of the following measures was required within the workcenter:

- o time-in-queue
- o time-in-transport
- o sizes of WIP
- o utilization of equipment
- o proportion of time a machine is broken, blocked, or starved
- o proportion of jobs produced that must be reworked or scrapped.

These performance measures could be used within other workcenters and not preclude a metamodel for the current one. In essence, the metamodeling decision must be evaluated on a workcenter by workcenter basis.

The second question deals with the capabilities of the OOM advanced modeling environment. The current implementation of OOM does not readily provide information in support of the following subset of the ten performance measures:

- o proportion of time that a machine is broken, blocked, or starved
- o proportion of jobs produced that must be reworked or scrapped
- o return on investment of a new or modified manufacturing system
- o payback period of a new or modified manufacturing system.

¹It is certainly conceivable that the distribution of a performance measure other than time-in-system could be the output of a metamodeling exercise.

While it might be possible to extract information useful to these types of analyses from the current implementation, they are not directly supported.

The metamodeling amenability question now becomes quite mechanical. Each workcenter is checked to see if its needed performance measures concur with the metamodeling aggregation and to see if the OOM environment supports the performance measures. If the answer to both questions is "yes", then the workcenter is a candidate for metamodeling. Table IX summarizes this candidate assessment process.

TABLE IX
METAMODELING CANDIDATE ASSESSMENT

WORKCENTER PERFORMANCE MEASURE	CONCURS WITH AGGREGATE PERFORMANCE	SUPPORTED BY OOM	METAMODELING CANDIDATE
Part/Job Throughput	YES	YES	YES
Time-In-System for Parts/Jobs	YES	YES	YES
Time-In-Queue for Parts/Jobs	NO	YES	NO
Time-In-Transport for Parts/Jobs	NO	YES	NO
Sizes of WIP Inventory	NO	YES	NO
Utilization of Equipment/Personnel	NO	YES	NO
Proportion of Time that a Machine is Broken, Blocked, or Starved	NO	NO	NO
Proportion of Jobs Produced that must be Reworked or Scrapped	NO	NO	NO
Return On Investment of a New or Modified Manufacturing System	YES	NO	NO
Payback Period of a New or Modified Manufacturing System	YES	NO	NO

Availability Of A Metamodel

The second phase of the selection procedure is the determination of whether a plug-compatible metamodel is available. This phase is performed internally within the advanced modeling environment. The analysis is carried out any time the user highlights a candidate workcenter and selects the "Meta Replace" option in the plant definition window.

As stated in the Introduction, the metamodel availability question is based on (1) the workcenter structure (i.e., physical layout and routings), (2) the input to the workcenter (i.e., ratio of arrival rate to service rate), and (3) the workcenter operating characteristics (i.e., alternate routings, breakdowns, multiple concurrent resource requirements, etc.). The metamodel development procedure proposed in Chapter V and evaluated in Chapter VIII is designed to create a workcenter metamodel that is valid over a range of input levels. On the contrary, the metamodel is valid only for the structure and operating characteristics for which it was built.

When the user loads a "metamodel file" into the OOM environment, its name (i.e., the name of the workcenter) is stored in a list of available metamodels. The environment does not (at the current time) maintain any information relative to the structure and operating characteristics of the workcenter from which the metamodel was built.² Thus, the user is obligated to delete a metamodel that has become invalid due to a change of this type. If the change is temporary or experimental, rather than deleting the metamodel, the user could simply choose not use it until it is again valid.

With regard to the range of input levels over which a metamodel is available, the OOM environment provides direct feedback. The feedback comes in the form of upper and lower bounds for the metamodel. When a "metamodel file" is loaded, it explicitly carries with it the range of input values over which it has been studied. If an input level

²Maintaining and monitoring this information is certainly conceivable since the majority of the necessary information is stored in the "filed out" versions of the plant, BOM, and routing definitions.

is subsequently specified that is not between the bounds, a message is displayed which suggests that the metamodel is not valid at the specified level. The user must then use the base model of the workcenter rather than the metamodel.

In summary, the process of evaluating metamodel availability involves the OOM environment answering the following two questions:

- o is the selected workcenter name on the metamodel list?
- o is the specified input value within the bounds of the metamodel?

If the answer to both questions is "yes" then a plug compatible metamodel is available.

The question of judging its validity is the subject of the next chapter.

CHAPTER VIII

EVALUATION OF THE METHODOLOGY

Introduction

This chapter presents the evaluation of the proposed methodology. The evaluation is accomplished by implementing the methodology outlined in Chapter V within the object oriented framework presented in Chapter VI.

Experimental Evaluation

The experimental evaluation was conducted through a series of six scenarios. Each experimental scenario corresponds to one of the workcenter scenarios presented in Chapter V. Table X on the next page provides a list of the scenarios and the corresponding scenario IDs that are used throughout the remainder of this dissertation. Scenarios QN1 and QN2 are independent of each other and of all of the observation-based scenarios. Scenarios OB1 through OB4 are cumulative. Scenario OB2 starts with scenario OB1 and adds multiple concurrent resources, scenario OB3 starts with scenario OB2 and adds machine breakdowns, etc¹.

The process of evaluation for the first scenario, QN1, will be presented in detail. Subsequent scenarios will be presented in a more summarized fashion with any deviations from the QN1 process noted. In particular, the process for the third scenario, OB1, will be presented in more detail since it represents the first of the observation-based metamodels.

¹A slight deviation from this cumulative approach was required in scenario OB4 to accommodate "blocking". This deviation is discussed in the review of scenario OB4 results.

TABLE X
SCENARIO IDENTIFICATION

SCENARIO	DESCRIPTION	SCENARIO ID
Queueing Network Scenario 1	Tandem Network	QN1
Queueing Network Scenario 2	Tree Network	QN2
Observation-Based Scenario 1	State Dependent Routings	OB1
Observation-Based Scenario 2	Multiple Concurrent Resources	OB2
Observation-Based Scenario 3	Machine Breakdowns	OB3
Observation-Based Scenario 4	Finite Queues	OB4

Characteristics Common To All Scenarios

Certain characteristics are common to all the scenarios considered in this research. Perhaps the most important of these shared characteristics is that the workcenters to which metamodeling is applied all have balanced stages, in that the mean throughput capacity of each stage is equal. For example scenario QN1 has three stages each composed of a single machine whose throughput is one part per time unit. Scenario OB1 has two stages. Stage one has one machine with a throughput of one part per time unit. Stage two has three parallel alternate machines each with a throughput of 1/3 part per time unit. The importance of this characteristic is that it allows a single parameter (ρ) to be used as the approximate utilization value for each stage and therefore, the entire workcenter.

A second important characteristic shared by all workcenters in the plant validation runs (not just the metamodeled workcenters) is that they are stable. In this context, stable means that the mean arrival rate is less than the mean service rate for all stages. This requirement (a common one for queueing analysis) ensures that the queues

will not grow infinitely large. As an additional restriction for this research, heavy traffic situations ($\lambda/\mu > 0.80$) are not considered. Although the metamodeling methodology presented below is applicable to such systems, the development and validation run lengths as well as number of repetitions would undoubtedly have to be much greater to obtain reasonable estimates of the time-in-system distributions.²

Another shared characteristic of all workcenters in the plant validation runs is that they are conservative. In this context, conservatism means that parts are not lost from the system. Thus, parts cannot balk; any part that enters the system will eventually be processed and exit the system.

The importance of stability and conservatism is that, together with the structure and routings of the workcenters, the mean arrival rate to the metamodel can be determined by flow balance relationships. This allows the parameter associated with the metamodel (i.e., mean arrival rate divided by stage service rate) to be determined in a static a priori way rather than dynamically. This principle, along with the steady state assumption, is what preempted the need for the dynamic parameterization procedure originally proposed in Chapter V. This procedure was not developed simply because it was not needed.³

Scenario QN1 - Tandem Queueing Network

Introduction

Queueing network scenario one is a tandem queueing network composed of three stages (machines). Parts enter the workcenter and are routed sequentially to each machine and then exit the workcenter. The three service time distributions are

²Special thanks go to committee member Dr. Manjunath Kamath for his recognition of this fact at the proposal phase of this research.

³A second special thanks to committee member Dr. Manjunath Kamath for his recognition of the fact that the metamodels in this research are steady state, not dynamic, models.

independent identically distributed (iid) exponential random variables with a mean ($1/\mu$) of one time unit. The arrival process is Poisson, therefore the interarrival time of parts to the workcenter is exponentially distributed. Since one of the purposes of the experiments is to test (and validate) the workcenter metamodels over a range of values of stage utilizations, the mean of the interarrival time distribution ($1/\lambda$) is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted utilization. The corresponding stage utilization values (λ/μ) are thus 0.25, 0.40, 0.60, and 0.80. Figure 18 illustrates this scenario.

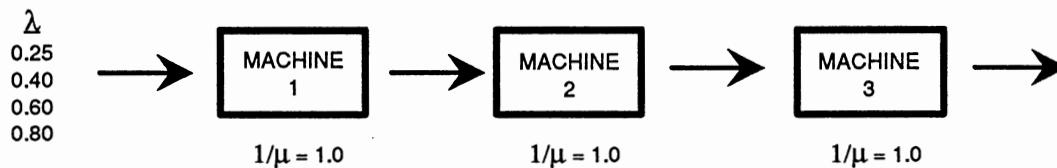


Figure 18. Workcenter QN1 Structure

One of the first major issues to be addressed concerned the design characteristics of simulation runs required to develop a metamodel. Remembering that the objective of the metamodel is to approximately model the entire distribution of part time-in-system, these experimental design questions included:

- o How is "to approximately model" to be judged?
- o How long must each simulation run be (measured in number of collected time-system values) to ensure that the time-in-system distribution (particularly the distribution "tails") are approximately modeled?

- o How long should the "warm-up" period (measured in time units) be to eliminate the idle and empty start-up influence on the collected time-in-system statistics?
- o How many simulation runs at each utilization value are required to ensure that the time-in-system distribution (particularly the distribution "tails") are approximately modeled?
- o How should random deviates be generated for each simulation run (i.e., single or multiple streams, using what random number generator, using what seed values)?

These issues were resolved using an empirical approach involving the examination of reasonable (in the author's judgement) potential values. This empirical approach and its validation is discussed in Appendix B. The evaluation resulted in the following conclusions which were used throughout the remainder of the experimentation:

- o "Approximately modeled" is judged adequate if, for a given stage utilization value, the average time-in-system cumulative distribution function (CDF) curve obtained from the metamodel simulation runs does not violate the Kolmogorov-Smirnov goodness of fit limits [Massey 1951] obtained from the average time-in-system CDF curve from the base model simulation runs.
- o Each simulation model is run until 3,000 time-in-system values are collected after statistics have been cleared.
- o Statistics are cleared in each run after 300 time units (as points of reference, the greatest mean interarrival time for any model is four time units and the greatest mean time-in system for any model is 23.5944 time units).
- o Five simulation runs at each utilization value are used to calculate the average CDF.

- o Random deviates are generated using the inverse transformation method with Park and Miller's [1988] random number generator. A multiple stream approach is maintained with each stochastic process (i.e., machine M1 service time, Part interarrival time, etc.) maintaining its own random number stream. Five seeds are associated with each stochastic process (one for each of the five simulation runs), seeds are held constant for each process across model⁴, decision, and utilization rate variants. Seeds were randomly selected from a table of 5000 hand drawn random numbers [Mize and Cox 1968, 218-219].

For scenario QN1 the distributional form for the metamodel can be analytically determined since QN1 is an open queueing network with a product form solution. Walrand and Varaiya [1980] have shown that for Jacksonian networks with non-overtaking paths (of which this is one) the sojourn times of a customer in consecutive nodes are independent. It follows directly that if the distributional form of the sojourn times at each node are known, then the time-in-system distribution (i.e., the distribution formed by the sum of node sojourn times) can be calculated.

Ross [1989, 354-355] has shown that for an M/M/1 queueing model, a single server exponential queueing model with exponential interarrivals (rate λ) and exponential service (rate μ), the sojourn time distribution is exponential (rate $\mu - \lambda$). Further, Ross [1989, 278] has shown that the output of a stable ($\rho < 1$) M/M/1 queueing model is a Poisson process with rate λ . Scenario QN1 is a tandem network consisting of three M/M/1 queueing models. Drawing on Ross' results, the results of Walrand and Varaiya, and the fact that the sum of iid exponential random variables is a gamma random variable, it can be shown that the time-in-system random variable for scenario QN1 is gamma distributed with shape parameter three and scale parameter $\mu - \lambda$.

⁴Since the stochastic processes vary by model, the random number seeds in fact vary by model. This introduces a source of variation which must be accounted for in the ANOVA used for multiple model comparisons.

Plant Level Validation #1

As discussed in Chapter V, the plant level validation of the metamodels is conducted by including the metamodeled workcenter (in both its base and metamodel forms) within a plant model and evaluating several decision alternatives with the plant model. The validity of the metamodel is judged based on consistency between the base model and the metamodel across decision alternatives. This consistency is judged by three validation tests. Validation #1 is an empirical visual validation. Validations #2 and #3 are statistically grounded tests.

Validation #1 involves the visual comparison of a line representing the metamodel performance across the two decision alternatives and a line representing the base model performance across the same alternatives. While acknowledging that this is highly empirical, it helps to substantiate the performance of the metamodel on a pragmatic basis. Figure 19 on the next page is an illustration of the type of graph used for plant level validation #1.

In an ideal situation, the two lines in Figure 19 would coincide. This would indicate perfect agreement between the metamodel and the base model for the decision alternatives. The degree to which the two lines do not coincide is a measure of the "approximate" nature of the metamodel. Any differences that exist between the lines can be segregated into two types of error, an error due to inaccuracy and an error due to inconsistency. Both types of error are illustrated in Figure 19.

In terms of the geometry of lines, an error due to inaccuracy is an error in the y-intercept of the metamodel while an error due to inconsistency is an error in the slope. In a pragmatic sense, an error due to inaccuracy is an error in the "absolute" performance of the metamodel while an error due to inconsistency is an error in the "relative" performance. The main emphasis of this research is the relative performance of metamodels, therefore, the analysis will focus on errors due to inconsistency.

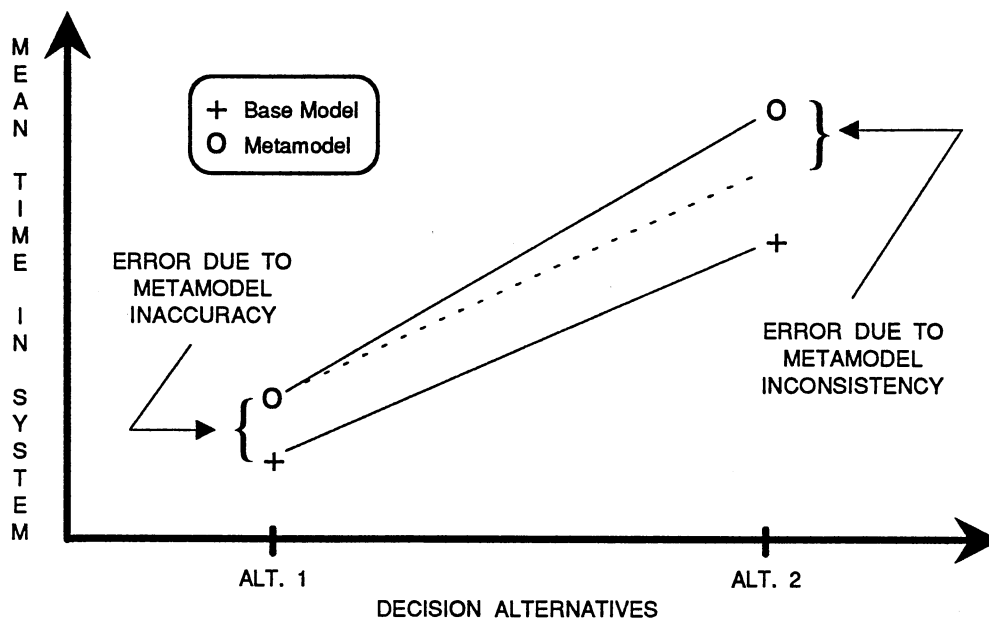
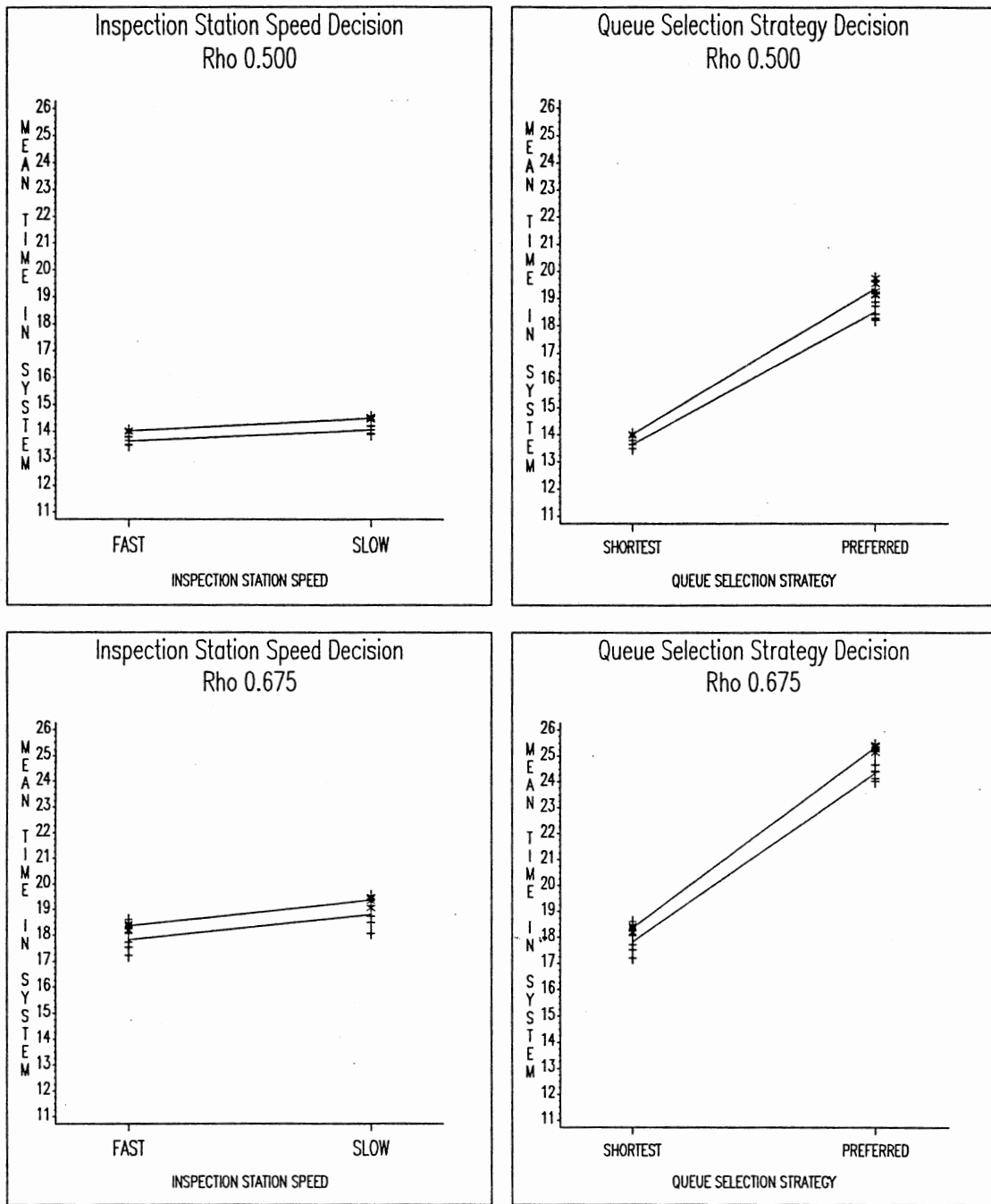


Figure 19. Plant Level Validation #1

For each scenario there are four plant level validation #1 graphs. The four graphs for scenario QN1 are shown in Figure 20 on the following page. Each graph represents one of the four possible combinations of the two decision cases (inspection station speed and queue selection strategy) and the two workcenter stage utilization values (ρ equals 0.500 and 0.675). In Figure 20, the two left hand graphs represent decision case I (inspection station speed) with ρ at 0.500 on top and ρ at 0.675 on bottom. The two right hand graphs represent decision case II (queue selection strategy) with ρ at 0.500 on top and ρ at 0.675 on bottom.

Each of the four graphs has a similar internal format. Each graph contains twenty plotted points and two lines. Each of the twenty plotted points represents a calculated mean time-in-system for a simulation run. The points can be categorized into four groups. Specifically, there are five points for each combination of the two decision



+ = Base Model <<>> * = Meta Model

PLANT QN1

Figure 20. Plant QN1 Validation #1 - Visual Inspection

alternatives (fast vs. slow inspection or shortest vs. preferred queueing) and the two model types (base vs. meta). In many cases the points so nearly coincide that they become indistinguishable. One line on the graph connects the mean of the metamodel runs operating under decision alternative one with the mean of the metamodel runs operating under decision alternative two. The other line is drawn similarly for the base model runs. The values of the means that the lines connect are not plotted on the graph.⁵

As stated above, the results of plant level validation #1 are empirical. Due to this subjective and judgmental nature the results below are stated in terms of observations rather than definitive conclusions. A visual inspection of Figure 20 yields the following generalized observations:

- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o the metamodel inconsistency (when apparent) is divergent rather than convergent;
- o the degree of inconsistency is greater for the queue selection strategy decision than for the inspection station speed decision.

Plant Level Validation #2

Plant level validation #2 is a statistical test of the consistency of the decision outcome between the base model and the metamodel. This validation test is conducted independently for each model type. The test is conducted in the following manner:

- o an analysis of variance (ANOVA) is conducted to determine the observed significance level (OSL) of the difference between the mean time-in-system

⁵For consistency of comparison, this page layout, internal format, and graph scaling are maintained for the remaining scenarios.

of the base model runs with fast inspection using rho of 0.500 and the base model runs with slow inspection using rho 0.500;⁶

- o the ANOVA is repeated for the three additional combinations of decision alternatives and rhos (fast vs.slow inspection using rho 0.675, shortest vs.preferred queueing using rho 0.500, and shortest vs.preferred queueing using rho 0.675);
- o all four of the above ANOVAs are repeated using the metamodel runs instead of the base model runs;
- o the results of the ANOVAs are compared across model types (base and meta) for consistency of the significance decision at $\alpha=0.05$ and $\alpha=0.01$.

The results of the ANOVAs for scenario QN1 are shown in Table XI on the next page. Inspection of this table reveals that the corresponding base models and metamodels are entirely consistent. In every case, both the base model and the metamodel show a significant difference between the mean time-in-system across the decision alternatives (i.e., the null hypothesis is rejected).

It is noteworthy in reviewing the table that not only is the null hypothesis rejected, in each case it is rejected with an extremely small OSL (≤ 0.0001). This is primarily due to the large sample size used within the simulation runs (3000 time-in-system values). The ramification of a large sample size (n) is that the variance of the mean of multiple runs is very small due to the inverse root effect of sample size ($1/\sqrt{n}$) in its calculation. With a sample size this large, very small differences between means of multiple runs will be detected as statistically significant. Therefore, it is not surprising in Table XI that the null hypothesis is consistently rejected with a very small OSL.

⁶The statistical rationale and SAS programs for conducting this analysis of variance are provided in Appendix C.

TABLE XI
ANOVA SUMMARY FOR QN1 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
QN1/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN1/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN1/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN1/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN1/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN1/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN1/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN1/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0000	Reject	Reject

Plant Level Validation #3

The third and final plant level validation is a statistical test of the consistency of the decision effect across the base model and the metamodel. In terms of the validation #1 visual comparison (Figure 19), this test determines if a significant difference exists between the slopes of the base model line and the metamodel line. In statistical terms, the test determines the OSL of the interaction between the model type (base or meta) and the decision alternative (fast vs. slow inspection or shortest vs. preferred queueing).

The test is conducted in the following manner:

- o An ANOVA is conducted to determine the OSL of the model by decision interaction of the base model runs with fast versus slow inspection using rho of 0.500 and the metamodel runs with fast versus slow inspection using rho of 0.500.⁷

⁷The statistical rationale and SAS programs for conducting this analysis of variance are provided in Appendix C.

- o The ANOVA is repeated for the three additional combinations of decision alternatives and rhos (fast vs.slow inspection using rho 0.675, shortest vs.preferred queueing using rho 0.500, and shortest vs.preferred queueing using rho 0.675).
- o The results of the ANOVAs are examined to determine if any of the interaction effects are significant at $\alpha=0.05$ and $\alpha=0.01$.

The results of the ANOVAs for scenario QN1 are shown in Table XII on the next page. Inspection of this table reveals an interesting mixture of results. The desirable outcome of this test is for the statistical tests to show no difference in slopes between the associated base models and metamodels. This would allow a statistical conclusion of consistency to be drawn. Unfortunately, this occurred in only one of the four cases; fast versus slow inspection with rho at 0.675. Referring back to Figure 19, it is apparent that among the four graphs this one (the bottom left) has the least discernable difference in slopes of the lines.

In each of the other three cases, the slopes were determined to be statistically different at one or both of the significance (α) levels. While unfortunate from the standpoint of desired results, these conclusions are not totally unexpected. As before, the large sample sizes used in the individual simulation runs result in small differences being detectable as statistically significant. The larger question for this research effort becomes: does this statistically detectable difference represent a difference that is of practical consequence when dealing with a model that is, by intent and design, only an approximation?⁸

In an effort to pragmatically measure the differences in slope, an additional test was formulated and included as part of plant level validation #3 (henceforth this

⁸Appendix D presents several editorial perspectives on comparing two numbers. Thanks are extended to Dr. David Weeks for his help in defining these perspectives.

TABLE XII
ANOVA SUMMARY FOR QN1 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
QN1	Inspection Station Speed	0.500	$H_0: \Delta \text{Slope} = 0$	0.0020	Reject	Reject
QN1	Inspection Station Speed	0.675	$H_0: \Delta \text{Slope} = 0$	0.5735	Accept	Accept
QN1	Queue Selection Strategy	0.500	$H_0: \Delta \text{Slope} = 0$	0.0058	Reject	Reject
QN1	Queue Selection Strategy	0.675	$H_0: \Delta \text{Slope} = 0$	0.0302	Reject	Accept

validation is called plant level validation #3a). Referring back to Figure 19, the test compares the performance of the metamodel including the error due to inconsistency (the solid metamodel line) against the metamodel with the error due to inconsistency removed (the dashed line). The statistic that is calculated for this test is the magnitude of the error due to inconsistency expressed as a percent of the corrected metamodel value. In terms of the graph, this can be visualized as the difference between the right end points of the two metamodel lines divided by the right endpoint of the dashed metamodel line (expressed as a percentage).

The results of validation test #3a are shown in Table XIII on the next page. These results are considerably more satisfying from the standpoint of desirable outcome. An inspection of the table reveals that the worst case inconsistency error for a metamodel is 5.35 percent. Remembering that the metamodel is an approximate model, this level of error is, at least in the author's judgement, within the realm of acceptable performance. Of particular note, the level of error due to inconsistency would appear from the graphs of Figure 19 to be less in every case than the level of error introduced by metamodel inaccuracy.

TABLE XIII
ANOVA SUMMARY FOR QN1 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
QN1	Inspection Station Speed	0.500	14.50124	14.36667	0.94
QN1	Inspection Station Speed	0.675	19.35832	19.29664	0.32
QN1	Queue Selection Strategy	0.500	19.36968	18.38561	5.35
QN1	Queue Selection Strategy	0.675	25.32664	24.39712	3.81

Scenario QN2 - Tree Queueing Network

Introduction

Queueing network scenario two is a tree queueing network composed of three machines configured in two stages. Parts enter the workcenter and are routed to machine M1. After being serviced at M1, parts are randomly routed to either machine M2 or M3. After being serviced by either M2 or M3, parts exit the workcenter. The service time distribution for M1 is exponential with a mean ($1/\mu$) of one time unit. The service time distributions for M2 and M3 are iid exponentials with a mean of two time units ($2/\mu$). The arrival process to the workcenter is Poisson, therefore, the interarrival time of parts to the workcenter is exponentially distributed. The mean of the interarrival time distribution ($1/\lambda$) is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted stage utilization. The corresponding stage utilization values are thus 0.25, 0.40, 0.60, and 0.80. Figure 21 on the next page illustrates the workcenter structure of this scenario.

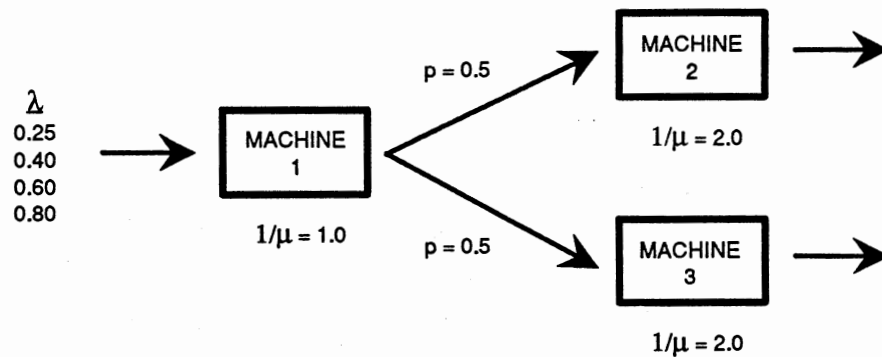


Figure 21. Workcenter QN2 Structure

Like scenario QN1, the distributional form for the metamodel for scenario QN2 can be analytically determined. The output from machine M1 is a Poisson process subject to splitting. It can be shown that a Poisson process with rate λ that is split with probability p forms two new Poisson processes with rates λp and $\lambda(1-p)$ [Ross 1989, 217-220]. For scenario QN2, this means that parts arrive at machines M2 and M3 according to a Poisson process with rate $\lambda/2$.

In summary, a part traversing workcenter QN2 encounters two stages. Stage one can be characterized as an M/M/1 queue with arrival rate λ and service rate μ . Stage two can be characterized as an M/M/1 queue with arrival rate $\lambda/2$ and service rate $\mu/2$. This network satisfies the non-overtaking path conditions of Walrand and Varaiya [1980], thus the time-in-system distribution for parts can be expressed as the sum of the stage sojourn time distributions.

Again using the results of Ross [1989, 354-355] for an M/M/1 queue, it can be shown that the stage sojourn time distributions are exponential with rate $\mu - \lambda$ for stage one and exponential with rate $\mu/2 - \lambda/2$ for stage two. The time-in-system distribution can now be expressed as the sum of two independent but non-identically distributed

exponential random variables (i.e., a hypoexponential distribution). Thus, the time-in-system distribution for parts in scenario QN2 is a two stage hypoexponential distribution with rates $\mu-\lambda$ and $\mu/2-\lambda/2$.

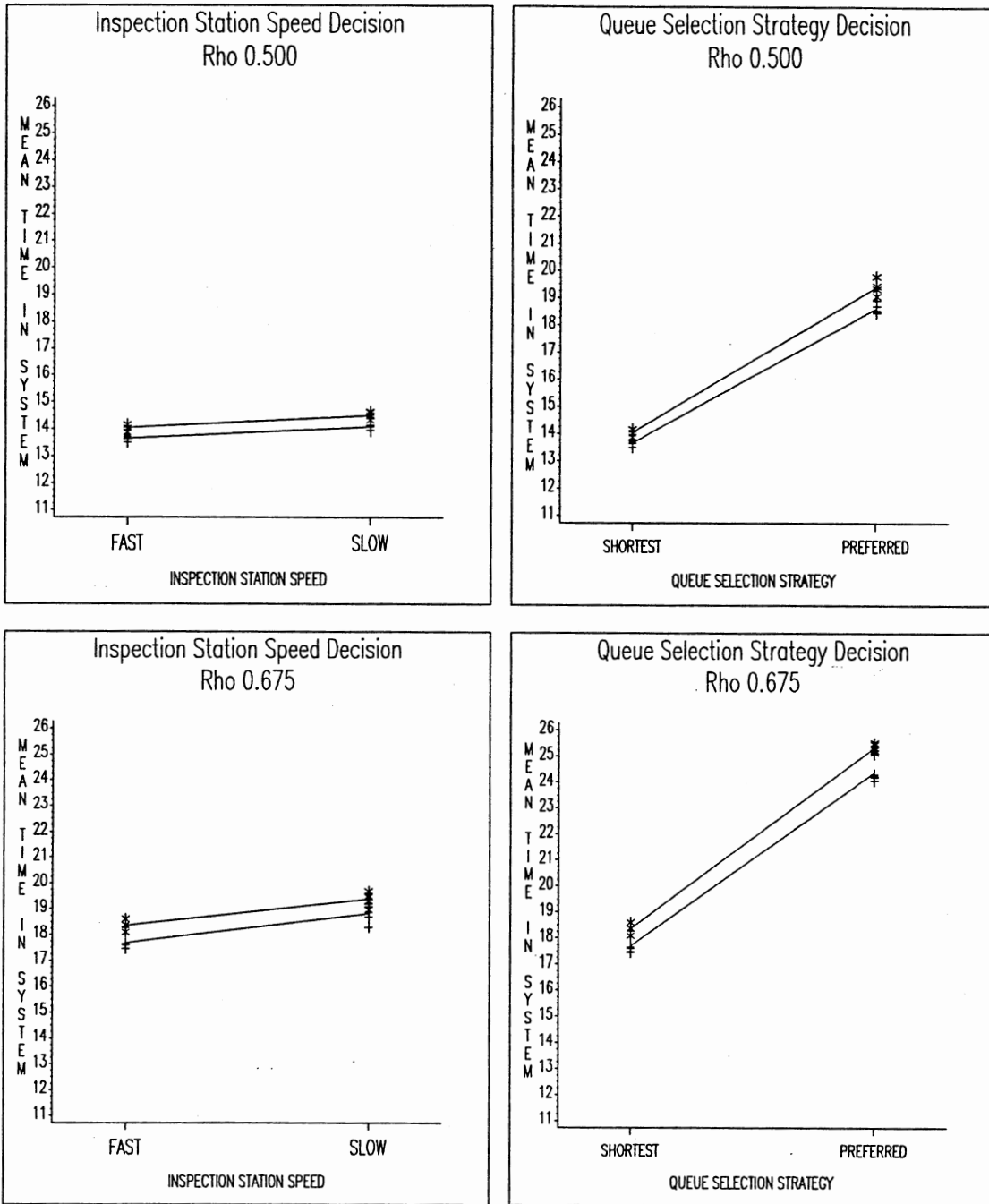
Plant Level Validation #1

The four plant level validation #1 graphs for scenario QN2 are shown in Figure 22 on the following page. An inspection of this figure yields observations similar to those for scenario QN1. Specifically:

- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o there is very little discernable inconsistency in the metamodels for the inspection station speed cases;
- o the inconsistency in the metamodels for the queue selection strategy cases is divergent rather than convergent;
- o the degree of inconsistency is greater for the queue selection strategy decision than for the inspection station speed decision.

Plant Level Validation #2

The results of the validation #2 ANOVAs for QN2 are shown in Table XIV on page 105. As in validation #1, the results are remarkably consistent with scenario QN1. In every case, both the base model and metamodel show a significant difference between the mean time-in-system across the decision alternatives.



+ = Base Model <<<>> * = Meta Model

PLANT QN2

Figure 22. Plant QN2 Validation #1 - Visual Inspection

TABLE XIV
ANOVA SUMMARY FOR QN2 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
QN2/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN2/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN2/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0047	Reject	Reject
QN2/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
QN2/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN2/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN2/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
QN2/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject

Plant Level Validation #3

The results of the statistical tests for validation #3 for QN2 are shown in Table XV on the next page. The results for the inspection station speed decision cases exhibit the desirable outcome in that the slopes of the base model and metamodel lines are not statistically different at either α value. However, for each of the queue selection strategy decision cases, the slopes are significantly different at both α values.

The above results are intuitively appealing upon a review of the graphs of Figure 22. The base model and metamodel lines in the left hand graphs (inspection station speed cases) are visually "very nearly" parallel. Contrarily, a noticeable difference in slopes can be seen in the right hand (queue selection strategy cases) graphs.

The results of the pragmatic validation test #3a are shown in Table XVI on the following page. These results are again considerably more satisfying from the standpoint of desirable outcome. An inspection of the table reveals that the worst case inconsistency

error for a metamodel is 4.20 percent. One interesting result of note in this table is that the base model line and metamodel line for the inspection station speed case with rho at 0.675 are convergent rather than divergent. This result is indicated by the negative percent error in the second line of the table.

TABLE XV
ANOVA SUMMARY FOR QN2 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
QN2	Inspection Station Speed	0.500	$H_0: \Delta Slope = 0$	0.0887	Accept	Accept
QN2	Inspection Station Speed	0.675	$H_0: \Delta Slope = 0$	0.5993	Accept	Accept
QN2	Queue Selection Strategy	0.500	$H_0: \Delta Slope = 0$	0.0054	Reject	Reject
QN2	Queue Selection Strategy	0.675	$H_0: \Delta Slope = 0$	0.0019	Reject	Reject

TABLE XVI
ANOVA SUMMARY FOR QN2 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
QN2	Inspection Station Speed	0.500	14.52261	14.41660	0.74
QN2	Inspection Station Speed	0.675	19.38936	19.61518	-1.15
QN2	Queue Selection Strategy	0.500	19.39344	18.61183	4.20
QN2	Queue Selection Strategy	0.675	25.33877	24.77294	2.28

Scenario OB1 - State Dependent Routings

Introduction

Observation based scenario one is a tree network composed of four machines configured in two stages. Parts enter the workcenter and are routed to machine M1 (stage one). After being serviced at M1, parts are routed to either machine M2, M3, or M4 (stage two). After being serviced by either M2, M3, or M4, parts exit the workcenter. The question of which stage two machine a particular part moves to is resolved on a state dependent basis. When a part's M1 processing is complete, the length of the input queue at each of stage two machines is examined and the part is routed to the machine with the shortest queue⁹. The service time distribution for M1 is triangular with a minimum, mode, and maximum of 0.9, 1.0, and 1.1 time units, respectively (henceforth expressed in the form TRI(0.9, 1.0, 1.1)). The service time distributions for M2, M3, and M4 are iid TRI(2.7, 3.0, 3.3). The arrival process to the workcenter is Poisson, therefore, the interarrival time of parts to the workcenter is exponentially distributed. The mean of the interarrival time distribution is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted stage utilization. The corresponding stage utilization values are approximately 0.25, 0.40, 0.60, and 0.80. Figure 23 on the next page illustrates the workcenter structure of this scenario.

The distributional form for the metamodel for this workcenter cannot be determined analytically. The presence of state dependent routings violate the product form solution assumptions required for an analytical solution to be calculable.

⁹Note that it is strictly the length of the input queue which is examined, the status of the machine (i.e., busy or idle) is not considered. In retrospect, inclusion of the machine status would have been a superior approach.

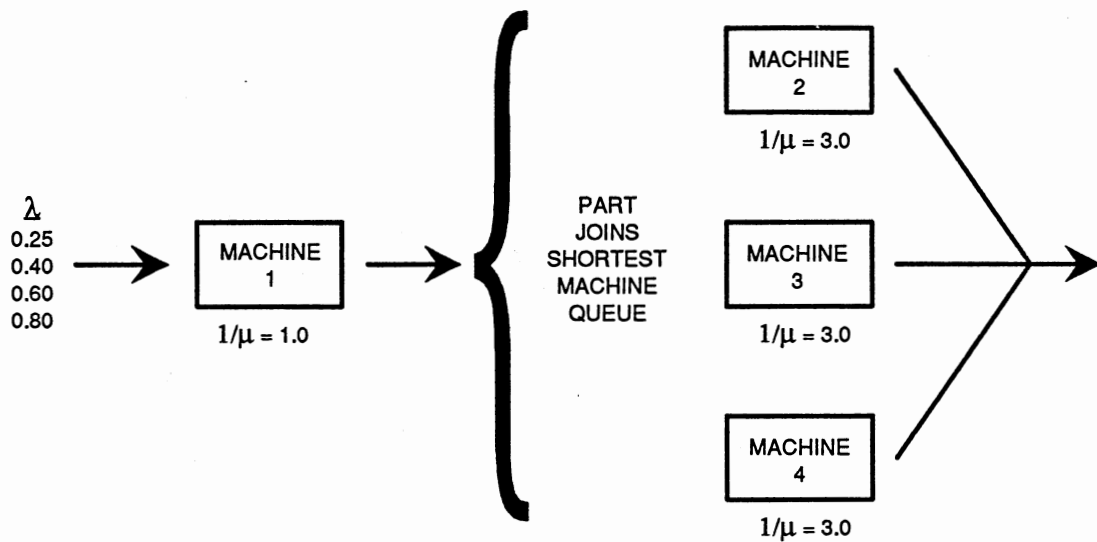


Figure 23. Workcenter OB1 Structure

Metamodel Development

As proposed in Chapter V, the metamodel development, usage, and validation procedures used in this research are founded on an observation-based methodology. The mechanics of the development methodology are as follows:

- o A simulation model of the workcenter is built (or extracted from the modeling database).
- o For each stage utilization value (0.25, 0.40, 0.60, and 0.80), five simulation runs are conducted and on each run 3,000 time-in-system observations are collected to be used in estimating the time-in-system distribution¹⁰.
- o For each combination of utilization value and simulation run (4 utilization values X 5 runs = 20 combinations), calculate an empirical grouped

¹⁰Refer to the discussions in (1) the Introduction to Scenario QN1 and (2) Appendix B for additional detail on the determination of simulation run characteristics.

cumulative distribution function (CDF) (refer to Appendix E for additional details on this procedure).

- o For each of the four utilization values, form an average empirical grouped CDF by arithmetically averaging the cell values of the five corresponding individual CDFs formed in the previous step.
- o Create a "metamodel file" (refer to Appendix F for a detailed specification of the file format).

Metamodel Usage

The mechanics of using a metamodel within the object oriented environment described in Chapter VI are automatically invoked upon selecting the "Metamodel Replace" option in the workcenter definition window. The steps of the usage methodology are as follows:

- o Select the workcenter within the plant model to be replaced by a metamodel.
- o Modify the plant definition by adding the metamodel version of the workcenter and deleting the base model components of the workcenter (note that the metamodel is automatically implemented with infinite servers since the metamodel sampling distribution already accounts for all queue time within the workcenter).
- o Modify the routing definition by replacing all base model operations in the metamodel workcenter with a single metamodel operation.
- o Determine (or estimate) the mean arrival rate (λ) to the metamodel.
- o Calculate the metamodel utilization parameter (ρ) by dividing λ by the mean stage service rate (μ) for the bottleneck stage of the base model version of the metamodel.
- o Supply the parameter ρ to the grouped empirical CDF service distribution of the metamodel workcenter (note if ρ is outside the range of stage utilization

values used to create the metamodel (for this research: $0.25 < \rho < 0.80$), this methodology does not support the use of the metamodel).

- o The parameter ρ is used to create a metamodel grouped empirical CDF via linear interpolation. The interpolation procedure uses the value arrays associated with the two closest stage utilization values found in the "metamodel file" for interpolation end points.
- o Whenever a sample is needed for the metamodel service time, the inverse transformation method is used to sample the interpolated grouped empirical CDF [Law and Kelton 1991, sec. 8.3.12].

Metamodel Validation

Metamodel validation is accomplished by comparing the results of simulation runs using the workcenter metamodel with results produced by the workcenter base model. Two stage utilization values (0.50 and 0.75) are used to validate the models. The mechanics of the validation process are as follows:

- o For a given stage utilization value, five simulation runs of the workcenter base model are conducted and on each run 3,000 time-in-system observations are collected to be used in estimating the time-in-system distribution.
- o For each simulation run, calculate an empirical grouped cumulative distribution function (CDF).
- o Form an average empirical grouped CDF by arithmetically averaging the cell values of the five individual CDFs formed in the previous step.
- o Calculate the Kolmogorov-Smirnov ($\alpha = 0.01$) goodness of fit limits [Massey 1951] for the average empirical grouped CDF.
- o For a given stage utilization value, five simulation runs of the workcenter metamodels are conducted and on each run 3,000 time-in-system observations are collected to be used in estimating the time-in-system distribution.

- o For each metamodel simulation run, calculate an empirical grouped cumulative distribution function (CDF).
- o Form an average empirical grouped CDF by arithmetically averaging the cell values of the five individual CDFs formed in the previous step.
- o Overlay the plot of the average empirical grouped CDF from the metamodel on a plot of the empirical grouped CDF from the base model with its associated goodness of fit limits.
- o If the metamodel CDF does not violate the goodness of fit limits over its entire range, then judge the metamodel valid; otherwise judge it invalid.
- o If the metamodel is invalid, execute the remedial procedure below.

Metamodel Remedial Procedure

The following procedure is designed to remedy the situation in which a metamodel has violated a goodness of fit limit during validation. In some cases the cycle of validation followed by the remedial procedure will need to be iterated several times before a successful validation is achieved. The remedial procedure, which is based on the interval halving search technique, is conducted as follows:

- o For the metamodel utilization values which failed validation (for instance $\rho = 0.75$), determine the utilization values that were used as endpoints in the implementation interpolation (for instance, $\rho = 0.60$ and $\rho = 0.80$).
- o Halve the interval represented by the interpolation endpoints and calculate new endpoints such that the target metamodel utilization value is at the midpoint of the halved interval (for instance, $\rho = 0.70$ and $\rho = 0.80$).
- o Execute the metamodel development procedure for either (or both) of the new endpoint utilization values for which the development procedure has not been previously run (for instance, $\rho = 0.70$).
- o Add this new grouped empirical CDF to the "metamodel file".

- o Re-execute the metamodel validation procedure (note that if the remedial procedure is required again for the parenthetical example being followed, the next set of new endpoints would be $\rho = 0.725$ and $\rho = 0.775$, both of which would require metamodel development runs).

Workcenter Level Validation

As proposed in Chapter V, the workcenter metamodels for the four observation based scenarios (OB1-OB4) are validated at stage utilization values (ρ) of 0.50 and 0.75. The results of these validations are presented in graphical form. For each scenario, four graphs are presented grouped into two figures. The first figure for each scenario presents the validation results for ρ at 0.50; the second for ρ at 0.75.

The top graph in each figure presents the probability density function (PDF) curve of the time-in-system distribution for both the base model and the metamodel. If the metamodel was perfectly accurate and consistent then these two curves would coincide.

The bottom graph in each figure presents the cumulative distribution function (CDF) curve of the time-in-system distribution for both the base model and the metamodel. Like the PDF graph, if the metamodel was perfectly accurate and consistent then these two curves would coincide. This graph also displays the upper and lower Kolmogorov-Smirnov ($\alpha = 0.01$) goodness of fit limits. A violation of either of these limits by the metamodel curve causes the metamodel to be judged invalid and the iterative cycling through the remedial procedure to be initiated. Since the results shown on the figures for each scenario are post-remedial, none of the metamodel curves on any of the graphs will violate the goodness of fit limits.

The SAS program used to produce the workcenter level validation graphs is shown in Appendix G. Besides producing the graphs that are used to visually inspect the validity of the workcenter metamodel, the program also quantitatively evaluates the

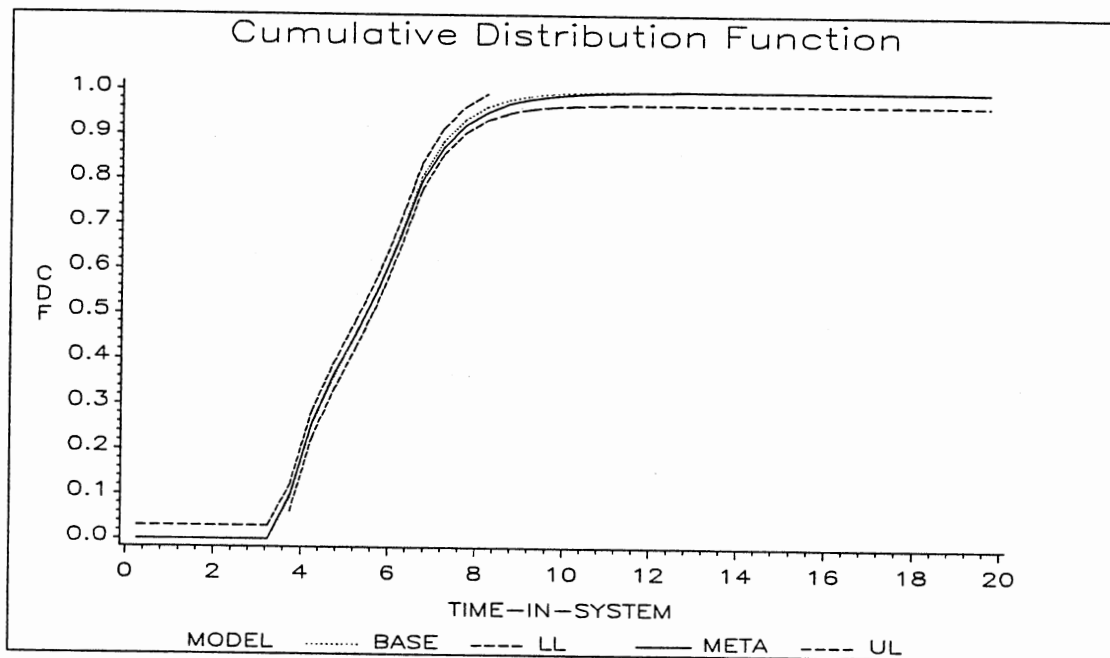
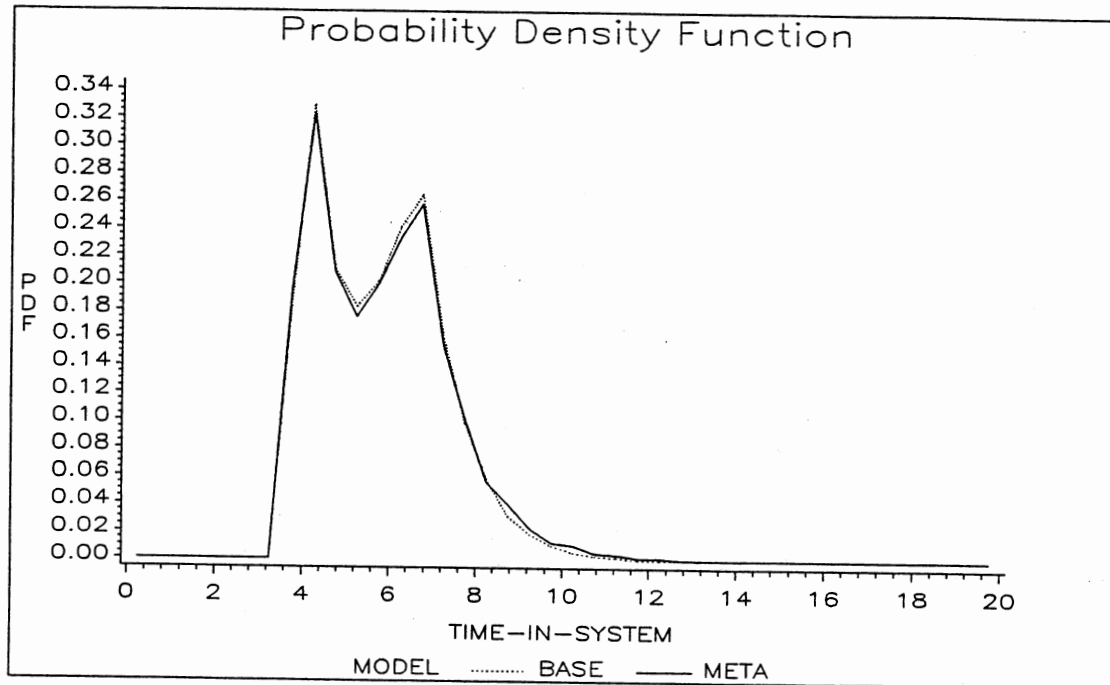
validity. At every cell along the CDF curve, the metamodel value is compared to the upper and lower goodness of fit limits. If any violations are detected the program terminates the validity check with a violation message. Results from this "go/no-go" test are not reported in a tabular fashion since the metamodels in the figures for each scenario are post-remedial, therefore, all the results would have been "go".

The workcenter validation graphs for scenario OB1 are shown in Figures 24 ($\rho=0.50$) and 25 ($\rho=0.75$) on the following two pages. In both cases, the valid metamodel presented in the graph was produced without resorting to remedial cycles. By observation, the $\rho=0.50$ metamodel appears to be comfortably within the goodness of fit limits while the $\rho=0.75$ metamodel appears to be quite close to the lower limit over a good deal of its range.

The PDF curves for both metamodels show an interesting bimodal characteristic. A review of other ρ values (not shown) reveals that this bimodal characteristic is consistent. The first mode is dominant at lower ρ values (approximately 0.50 and less) while the second mode is dominant at higher ρ values (greater than 0.50). At the extremes considered in this study, $\rho=0.25$ ($\rho=0.80$), the first (second) mode almost entirely subsumes the other mode.

It is the author's belief that this rather curious behavior can be at least partially accounted for with the following hypotheses:

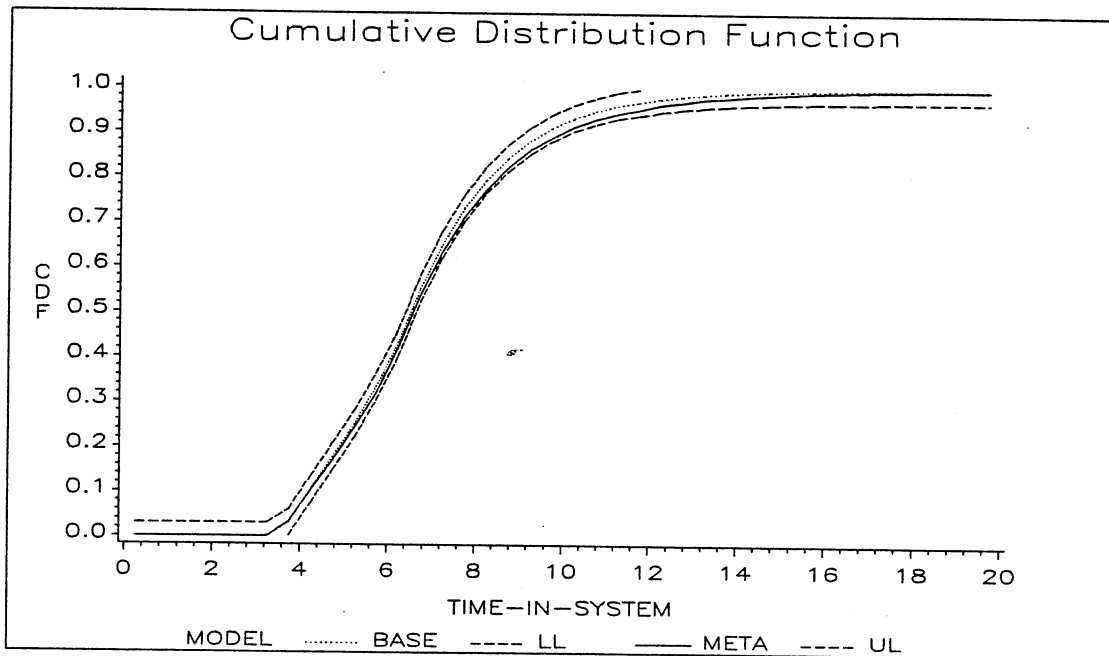
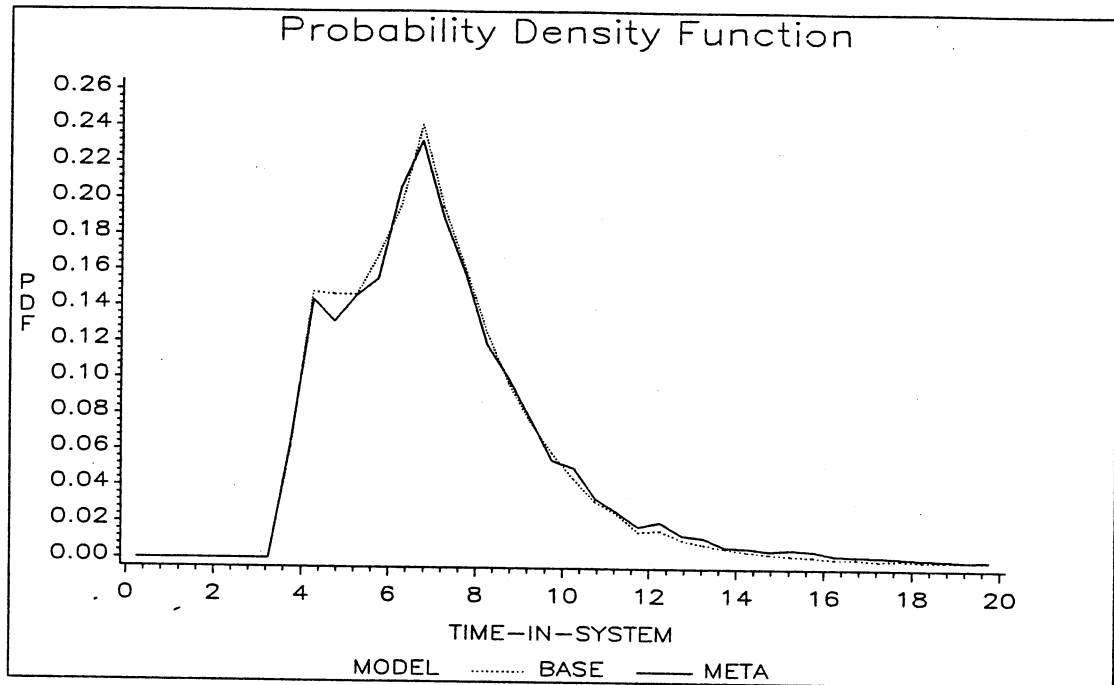
- o the first mode is representative of the time-in-system for parts that are queued to a stage two machine that is idle;
- o the second mode is representative of the time-in-system for parts that are queued to a stage two machine that is busy;
- o these two hypotheses are consistent with the observed changes in relationship between the two modes as ρ increases (i.e., as ρ increases, the probability of encountering an idle machine decreases and the second mode subsumes the first);



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB1 - Rho 0.50

Figure 24. Plant OB1 Workcenter Validation - Rho 0.50



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB1 - Rho 0.75

Figure 25. Plant OB1 Workcenter Validation - Rho 0.75

- o the sharp distinction between the modes is related to the shortest queue selection rule that ignores machine status. This hypothesis is supported by the fact that the bimodal nature of the time-in-system PDF is maintained through OB1, OB2, and OB3 but disappears in OB4 where the queue selection rule is modified to include machine status (among other changes).

Plant Level Validation #1

The four plant level validation #1 graphs for scenario OB1 are shown in Figure 26 on the following page. An inspection of this figure yields the following observations:

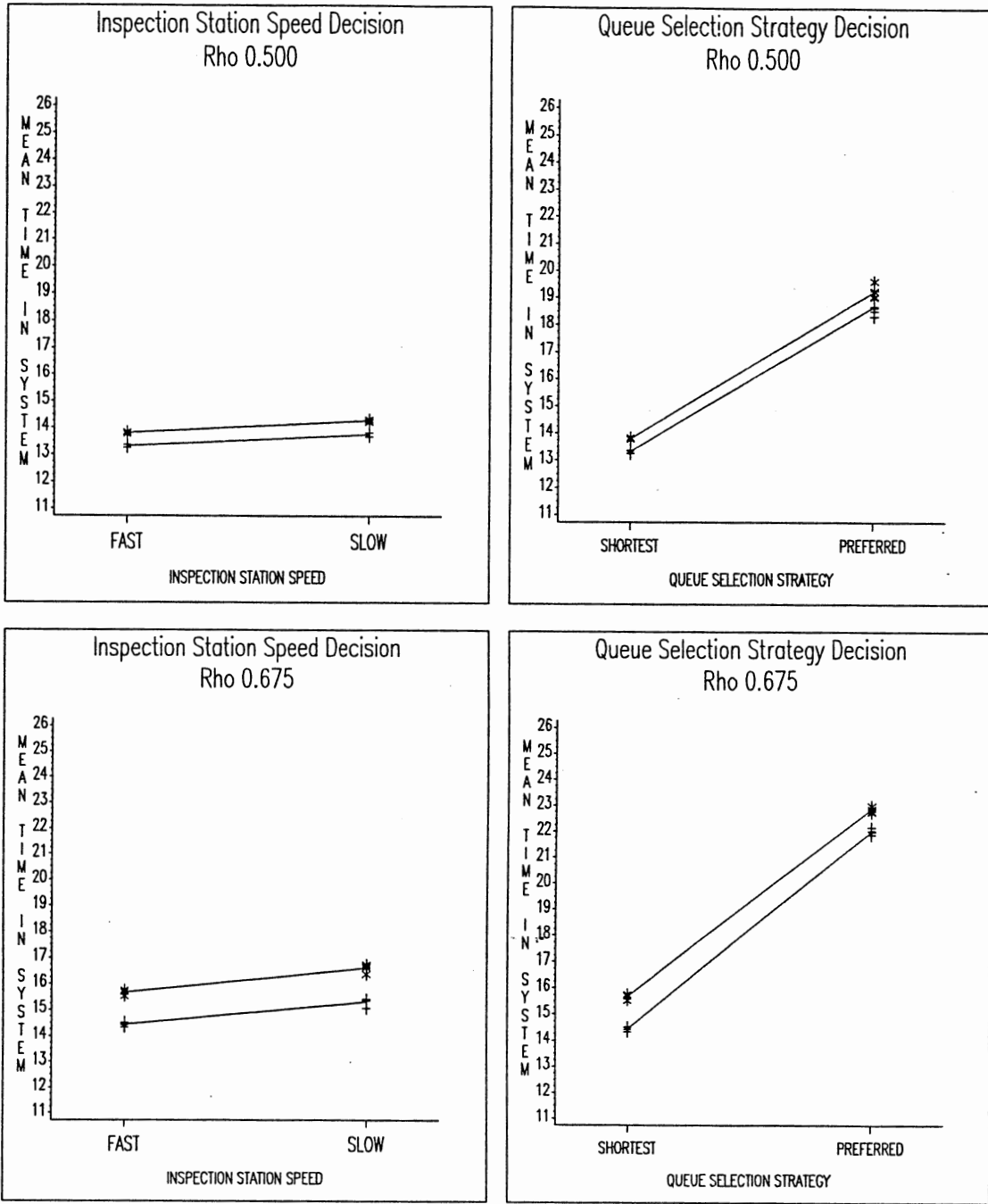
- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o the magnitude of the inaccuracies for both decision cases at rho 0.675 appears to be greater than in either of two previous scenarios;
- o there is very little discernable inconsistency in the metamodels. There does appear to be a slight convergence of the lines for the queue selection strategy decision at rho 0.675 (the lower right graph);

Plant Level Validation #2

The results for the validation #2 ANOVAs for scenario OB1 are shown in Table XVII on page 118. As in the prior scenarios, in every case a significant difference in mean time-in-system is detected across the decision alternatives.

Plant Level Validation #3

The results of the validation #3 ANOVAs for scenario OB1 are shown in Table XVIII on page 118. In all cases except the queue selection strategy case at rho 0.675 the table reflects the desired result (no difference in slope) at both α levels. In the



+ = Base Model <<<>> * = Meta Model

PLANT OB1

Figure 26. Plant OB1 Validation #1 - Visual Inspection

exceptional case, the difference in slope was detected with a very small OSL (0.0001). These results are intuitively appealing upon a review of Figure 26 since the lower right graph (the exceptional case) has the most perceptible slope difference.

TABLE XVII
ANOVA SUMMARY FOR OB1 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB1/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB1/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB1/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB1/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB1/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB1/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB1/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB1/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject

TABLE XVIII
ANOVA SUMMARY FOR OB1 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB1	Inspection Station Speed	0.500	$H_0: \Delta\text{Slope} = 0$	0.2783	Accept	Accept
OB1	Inspection Station Speed	0.675	$H_0: \Delta\text{Slope} = 0$	0.2663	Accept	Accept
OB1	Queue Selection Strategy	0.500	$H_0: \Delta\text{Slope} = 0$	0.7073	Accept	Accept
OB1	Queue Selection Strategy	0.675	$H_0: \Delta\text{Slope} = 0$	0.0001	Reject	Reject

The results of the pragmatic validation test #3a are given in Table XIX below. The worst case inconsistency is -3.27 percent (the negative shows the convergence noted in validation #1). In the other three cases, the results are highly desirable with each error being less than one percent.

TABLE XIX
ANOVA SUMMARY FOR OB1 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
OB1	Inspection Station Speed	0.500	14.26902	14.24311	0.18
OB1	Inspection Station Speed	0.675	16.61333	16.50004	0.69
OB1	Queue Selection Strategy	0.500	19.23814	19.10117	0.72
OB1	Queue Selection Strategy	0.675	22.86281	23.63557	-3.27

Scenario OB2 - Multiple Concurrent Resources

Introduction

Observation based scenario two is a tree network composed of four machines configured in two stages. The physical configuration and part routings are identical with those of scenario OB1 (see Figure 23). The difference between scenarios OB1 and OB2 is that, besides OB1's state dependent routing, OB2 adds a multiple concurrent resource requirement. In scenario OB2, a part must acquire both a machine and an operator to initiate processing at a machine. The operator completes a setup operation and then is

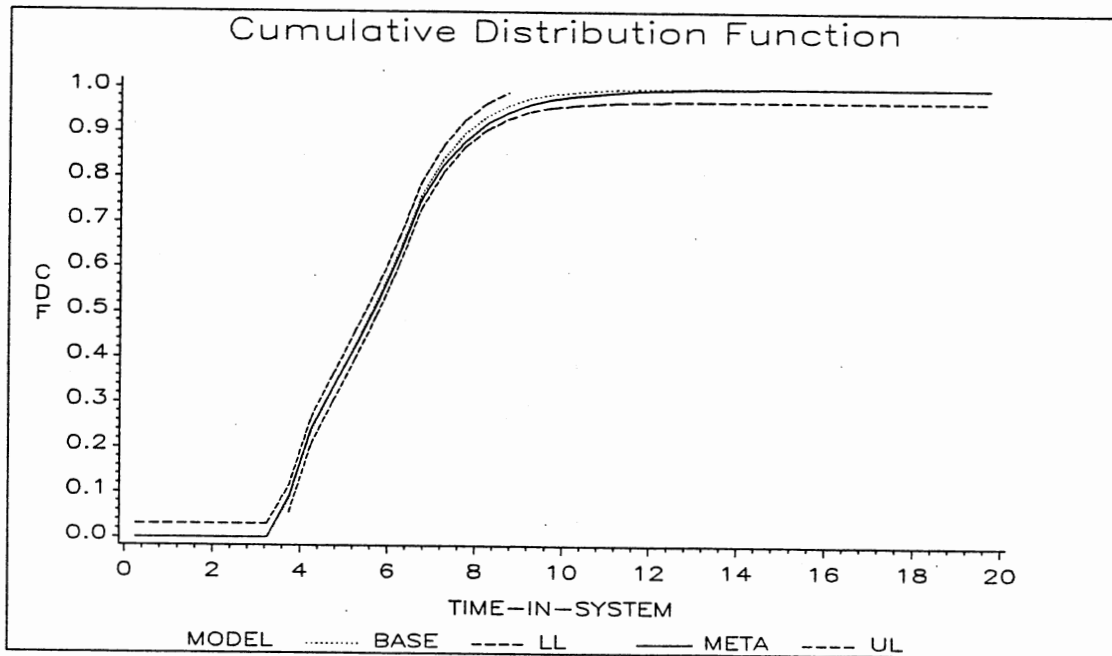
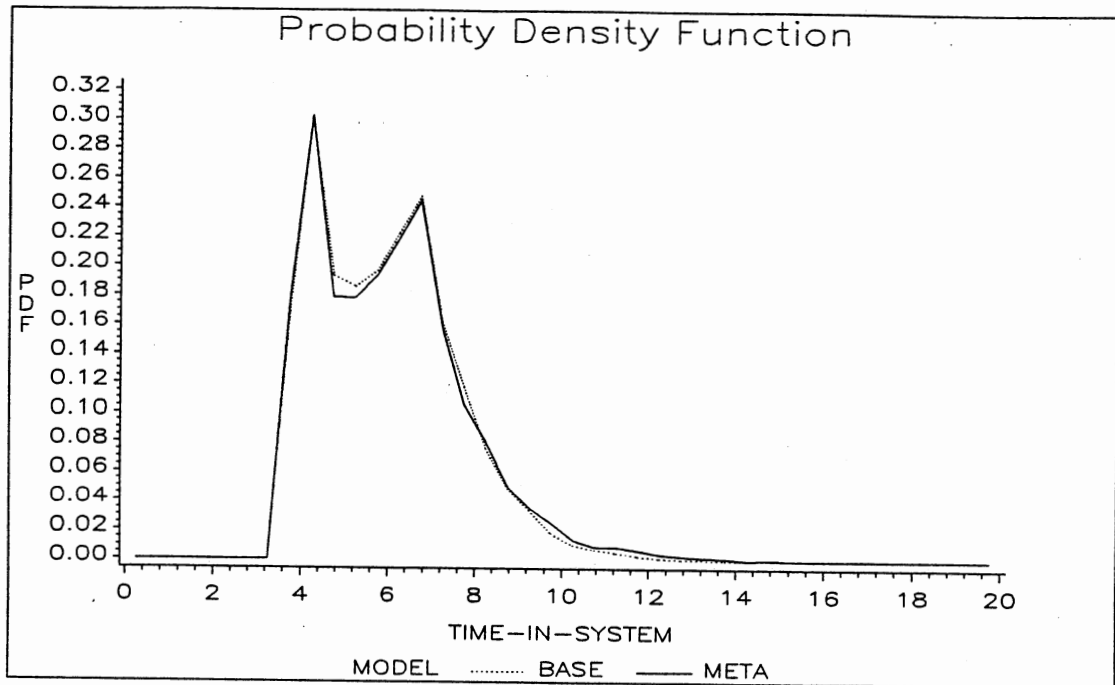
released. Only one operator is available within the workcenter to provide setup for all four machines (M1, M2, M3, and M4).

To maintain approximate consistency in the stage utilization values across the scenarios, the setup times for each operation (within workcenter OB2) are defined as the first ten percent of the scenario OB1 mean service time. The OB2 mean service time becomes ninety percent of the OB1 mean service time. The service time distribution for M1 thus becomes TRI(0.8, 0.9, 1.0) with a deterministic setup time of 0.1 time units. The service time distributions for M2, M3, and M4 become iid TRI(2.4, 2.7, 3.0) with deterministic setup times of 0.3 time units. The arrival process to the workcenter is Poisson, therefore, the interarrival time of parts to the workcenter is exponentially distributed. The mean of the interarrival time distribution is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted stage utilization. The corresponding stage utilization values are approximately 0.25, 0.40, 0.60, and 0.80.

The distributional form for the metamodel for this workcenter cannot be determined analytically. The presence of state dependent routings and multiple concurrent resources violate the product form solution assumptions required for an analytical solution to be calculable.

Workcenter Level Validation

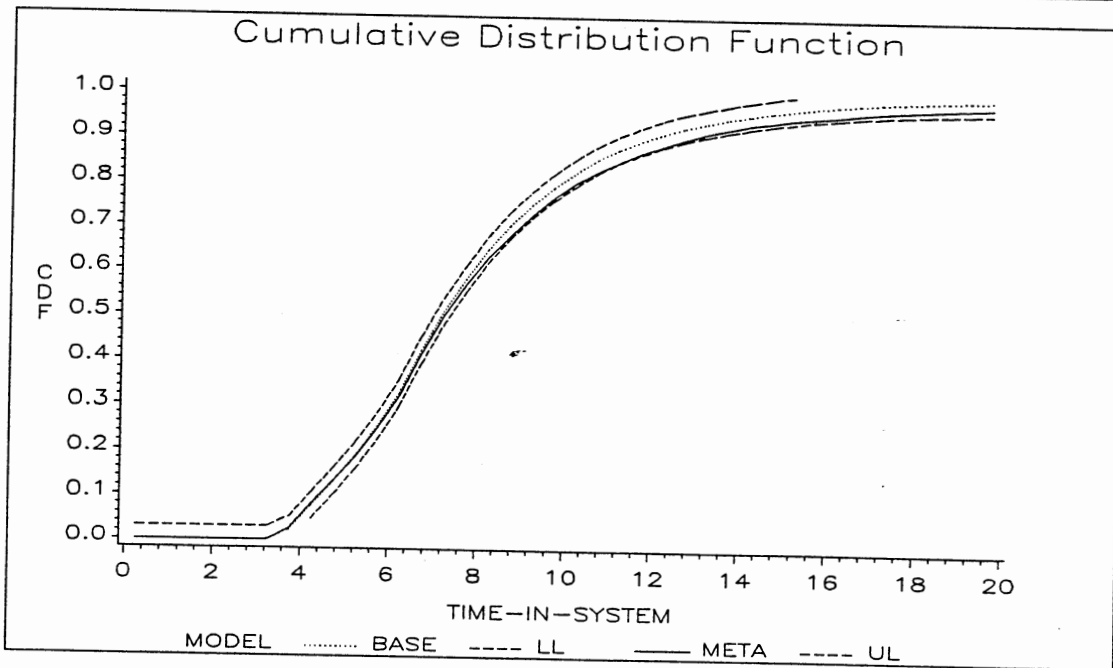
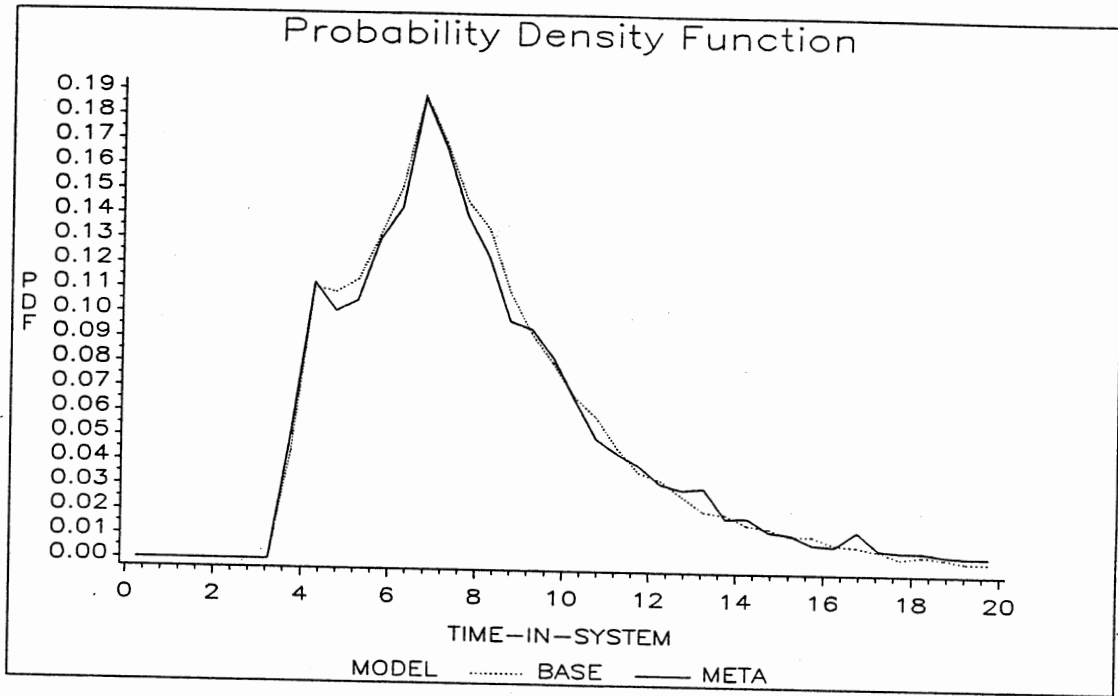
The workcenter validation graphs for scenario OB1 shown in Figures 27 ($\rho=0.50$) and 28 ($\rho=0.75$) on the following two pages. The metamodel with ρ at 0.50, Figure 27, was produced with no remedial cycles. Other than the valley between the modes, the metamodel PDF tracks the base model fairly well. For the CDF graph, there is a visible gap between the metamodel curve and base model curve between time-in-system values 7.0 and 12.0, but the entire metamodel curve is clearly within the Kolmogorov-Smirnov limits.



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB2 - Rho 0.50

Figure 27. Plant OB2 Workcenter Validation - Rho 0.50



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB2 - Rho 0.75

Figure 28. Plant OB2 Workcenter Validation - Rho 0.75

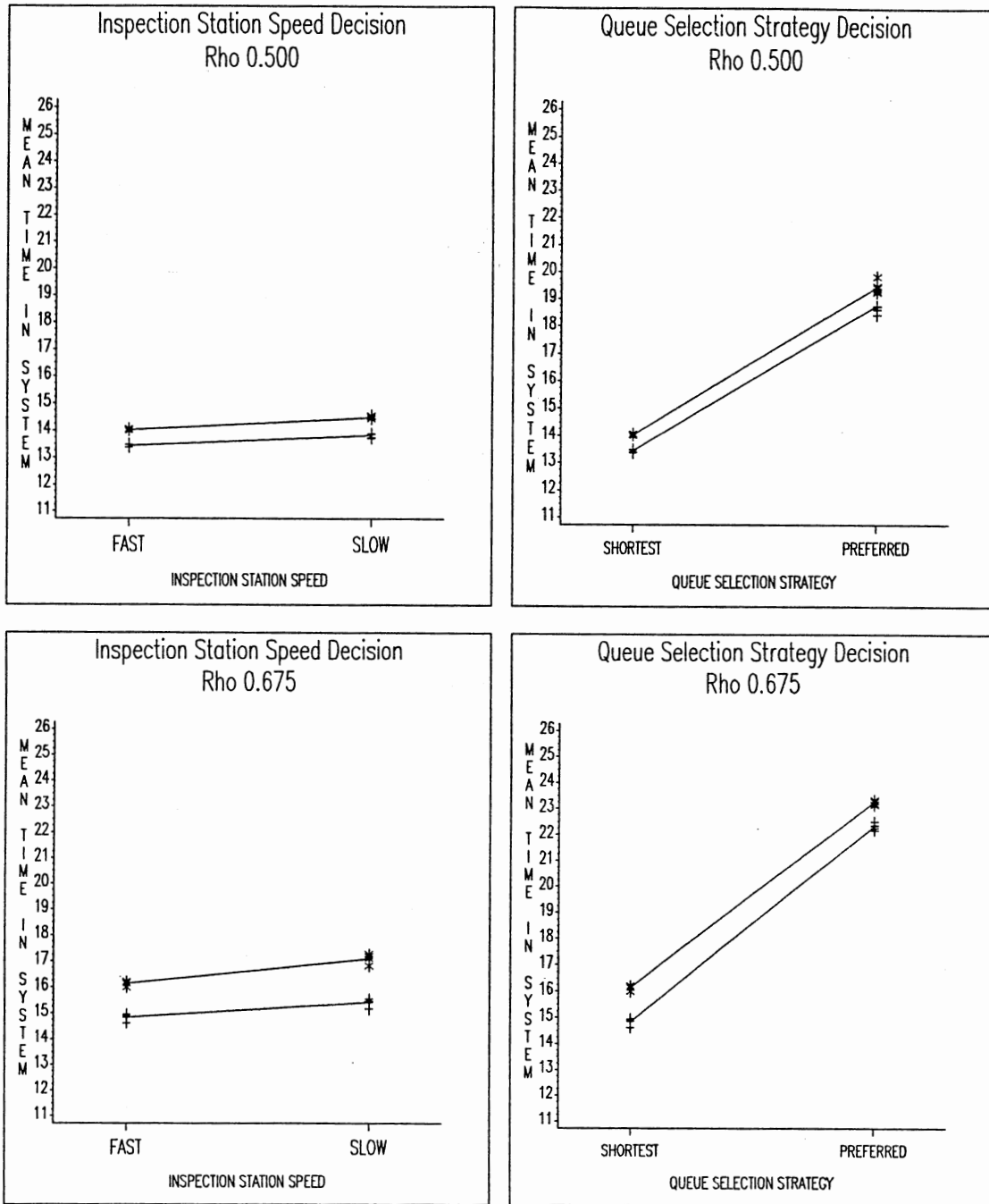
The metamodel with ρ at 0.75, Figure 28, required one iteration of the remedial procedure. This iteration caused the interpolation arrays to be associated with ρ values of 0.70 and 0.80 rather than 0.60 and 0.80. Even after the iteration, the metamodel CDF curve is only just passable in the time-in-system range 10.0 to 14.0. In fact, the quantitative rejection calculation performed by the SAS program of Appendix G was required to ensure that this curve was passable.

As in scenario OB1, the PDF curves for both metamodels show an interesting bimodal characteristic. A review of other ρ values (not shown) reveals that this bimodal characteristic is consistent. The first mode is dominant at lower ρ values while the second mode is dominant at higher ρ values.

Plant Level Validation #1

The four plant level validation #1 graphs for scenario OB2 are shown in Figure 29 on the next page. An inspection of this figure yields the following observations:

- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o the magnitude of the inaccuracies appears to be greater than either of the queueing network cases (QN1 and QN2) but consistent with the first observation based case (OB1);
- o there is very little discernable inconsistency in the metamodels with ρ at 0.500. With ρ at 0.675, the lines for the inspection speed decision appear to diverge. For the queue selection strategy decision they appear to converge slightly.



+ = Base Model <<<>> * = Meta Model

PLANT OB2

Figure 29. Plant OB2 Validation #1 - Visual Inspection

Plant Level Validation #2

The results for the validation #2 ANOVAs for scenario OB2 are shown in Table XX below. As in the prior scenarios, in every case a significant difference in mean time-in-system is detected across the decision alternatives.

TABLE XX
ANOVA SUMMARY FOR OB2 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB2/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB2/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB2/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB2/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB2/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB2/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB2/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB2/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject

Plant Level Validation #3

The results of the validation #3 ANOVAs for scenario OB2 are shown in Table XXI on the next page. Only for the queue selection strategy decision at rho 0.500 does the table reflect the desired result (no difference in slope) at both α levels. In all the other cases, a difference in slope was detected with a very small OSL (less than or equal to 0.0004). These results are not necessarily intuitively appealing upon a review of

Figure 29 since there does not appear to be a marked difference in slopes in any of the graphs. In particular, the lines on the inspection station speed decision at rho 0.500 appear very nearly parallel.

TABLE XXI
ANOVA SUMMARY FOR OB2 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB2	Inspection Station Speed	0.500	$H_0: \Delta Slope = 0$	0.0004	Reject	Reject
OB2	Inspection Station Speed	0.675	$H_0: \Delta Slope = 0$	0.0001	Reject	Reject
OB2	Queue Selection Strategy	0.500	$H_0: \Delta Slope = 0$	0.4799	Accept	Accept
OB2	Queue Selection Strategy	0.675	$H_0: \Delta Slope = 0$	0.0002	Reject	Reject

It is the author's contention that the resolution to this apparent dilemma lies in the variance of the plot points used to estimate the line slope. As the variance among the plot points at one (or both) ends of the line increases, the confidence interval around the slope estimate widens. This leads directly to an increased probability that the two slopes will not be considered significantly different. In terms of Figure 29, the plot points at the "preferred" end of both lines of the queue selection decision graph with ρ at 0.500 (upper right graph) have a greater variance among them than the points at the "slow" end of the other nearly parallel lines (upper left graph). Specifically, the calculated sample variances for the base and meta points on the upper right graph are 0.3180 and 0.2267 respectively while the same values for the upper left graph are 0.0804 and 0.0458.

The results of the pragmatic validation test #3a are given in Table XXII on the next page. The worst case inconsistency is an acceptable 4.37 percent. It is interesting

that for the queue selection strategy case with ρ at 0.675 the lines are converging as they were in scenario OB1. In every other case in both scenarios, the lines were diverging.

TABLE XXII

ANOVA SUMMARY FOR OB2 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
OB2	Inspection Station Speed	0.500	14.49896	14.31963	1.25
OB2	Inspection Station Speed	0.675	17.09285	16.37793	4.37
OB2	Queue Selection Strategy	0.500	19.45608	19.22343	1.21
OB2	Queue Selection Strategy	0.675	23.26270	24.05016	-3.27

Scenario OB3 - Machine Breakdowns

Introduction

Observation based scenario three is a tree network composed of four machines configured in two stages. The physical configuration and part routings are identical with those of scenario OB1 and OB2 (see Figure 23). The difference between scenarios OB2 and OB3 is that, besides OB2's state dependent routing and multiple concurrent resources, OB3 adds machine breakdowns. In scenario OB3, a machine can jam while it is processing a part. When a jam occurs, the operator is needed to un-jam the machine and restart the process. A jam cannot occur while a machine is idle. Only one operator is available within the workcenter to provide setup and service jams for all four machines (M1, M2, M3, and M4).

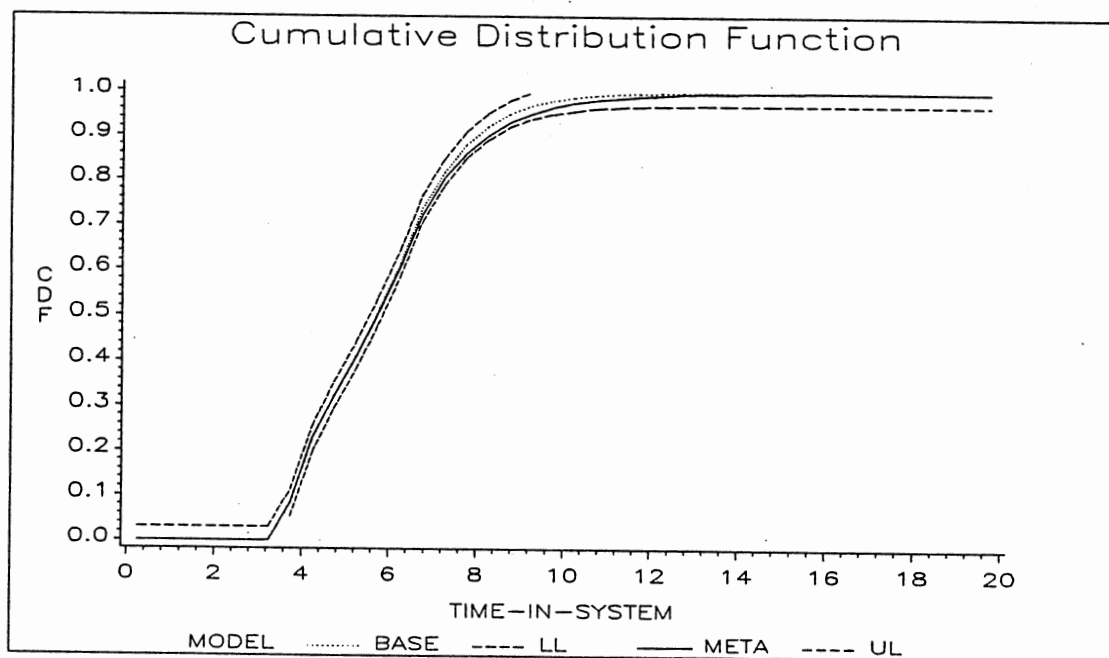
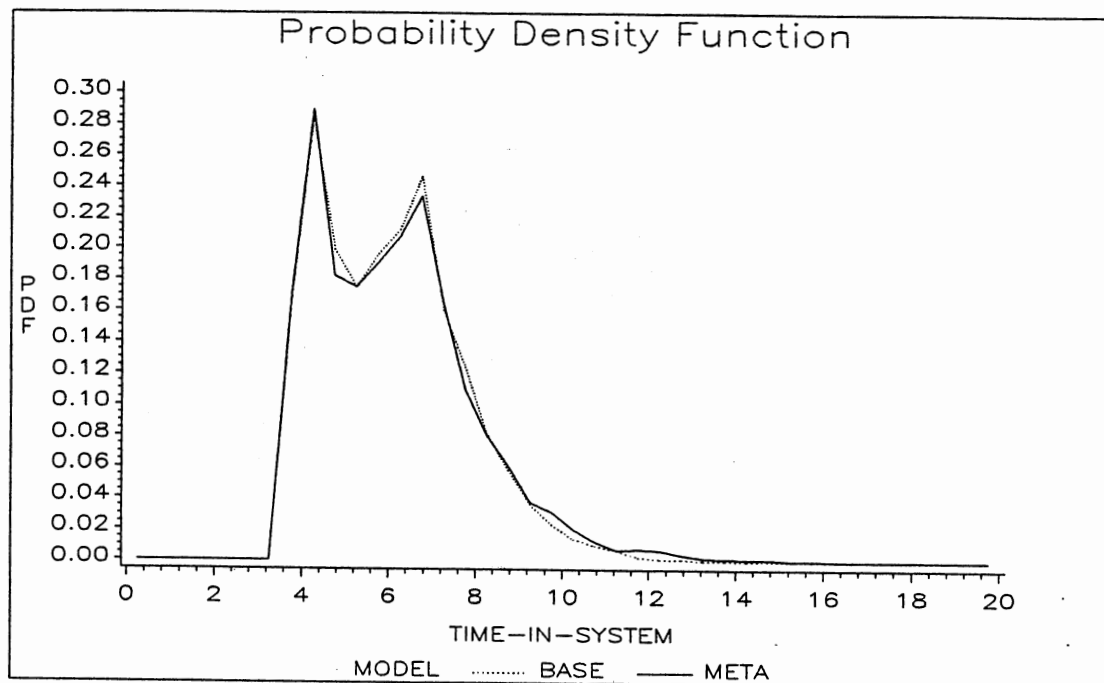
The service time and setup distributions for M1, M2, M3, and M4 remain as they are in scenario OB2. The machine failure and repair distributions for M1 are TRI(100, 100, 150) and TRI(0.5, 0.5, 1.0) respectively. The machine failure and repair distributions for M2, M3, and M4 are TRI(300, 300, 450) and TRI(1.5, 1.5, 3.0)¹¹. In both cases, when expressed in terms of the mean processing plus setup time for parts, the mean of the failure distribution is approximately once per 117 parts and the mean of the repair distribution is approximately 2/3 of a part's processing time. The arrival process to the workcenter is Poisson, therefore, the interarrival time of parts to the workcenter is exponentially distributed. The mean of the interarrival time distribution is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted stage utilization. The corresponding stage utilization values are approximately 0.25, 0.40, 0.60, and 0.80.

The distributional form for the metamodel for this workcenter cannot be determined analytically. The presence of state dependent routings, multiple concurrent resources, and machine failures violate the product form solution assumptions required for an analytical solution to be calculable.

Workcenter Level Validation

The workcenter validation graphs for scenario OB3 are shown in Figures 30 ($\rho=0.50$) and 31 ($\rho=0.75$) on the following two pages. The metamodel with ρ at 0.50, Figure 30, was produced with no remedial cycles. The metamodel PDF tracks well through the first mode but becomes more erratic through the valley and over the second mode. For the CDF graph, there is a visible gap between the metamodel curve and base

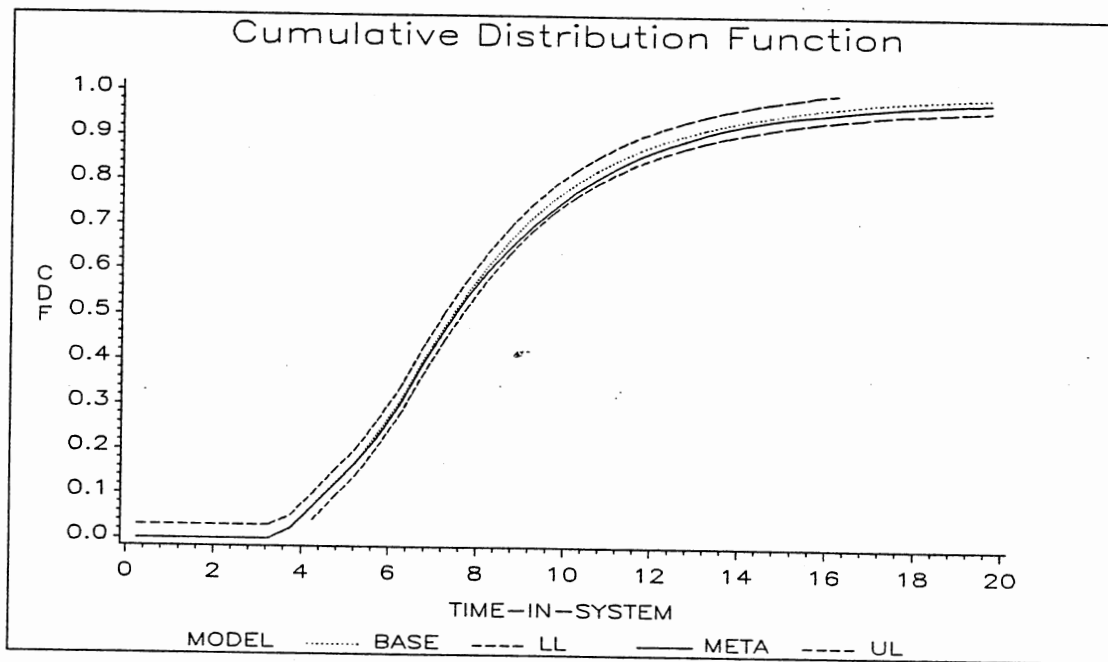
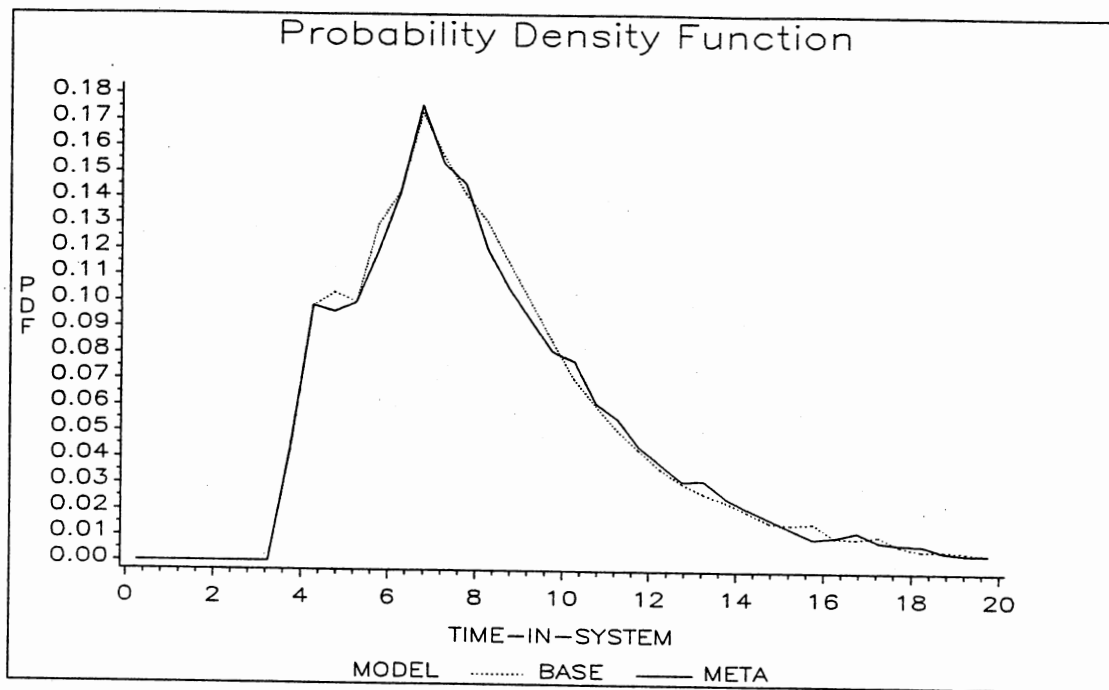
¹¹These distribution parameters were arrived at after considerable trial and error experimentation. The experimentation was designed to find parameters which would not overwhelm the system at high utilizations and yet still have some impact at low utilizations.



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB3 - ρ 0.50

Figure 30. Plant OB3 Workcenter Validation - ρ 0.50



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB3 - Rho 0.75

Figure 31. Plant OB3 Workcenter Validation - Rho 0.75

model curve over a broad range between time-in-system values 6.0 and 14.0, but the entire metamodel curve is clearly within the Kolmogorov-Smirnov limits.

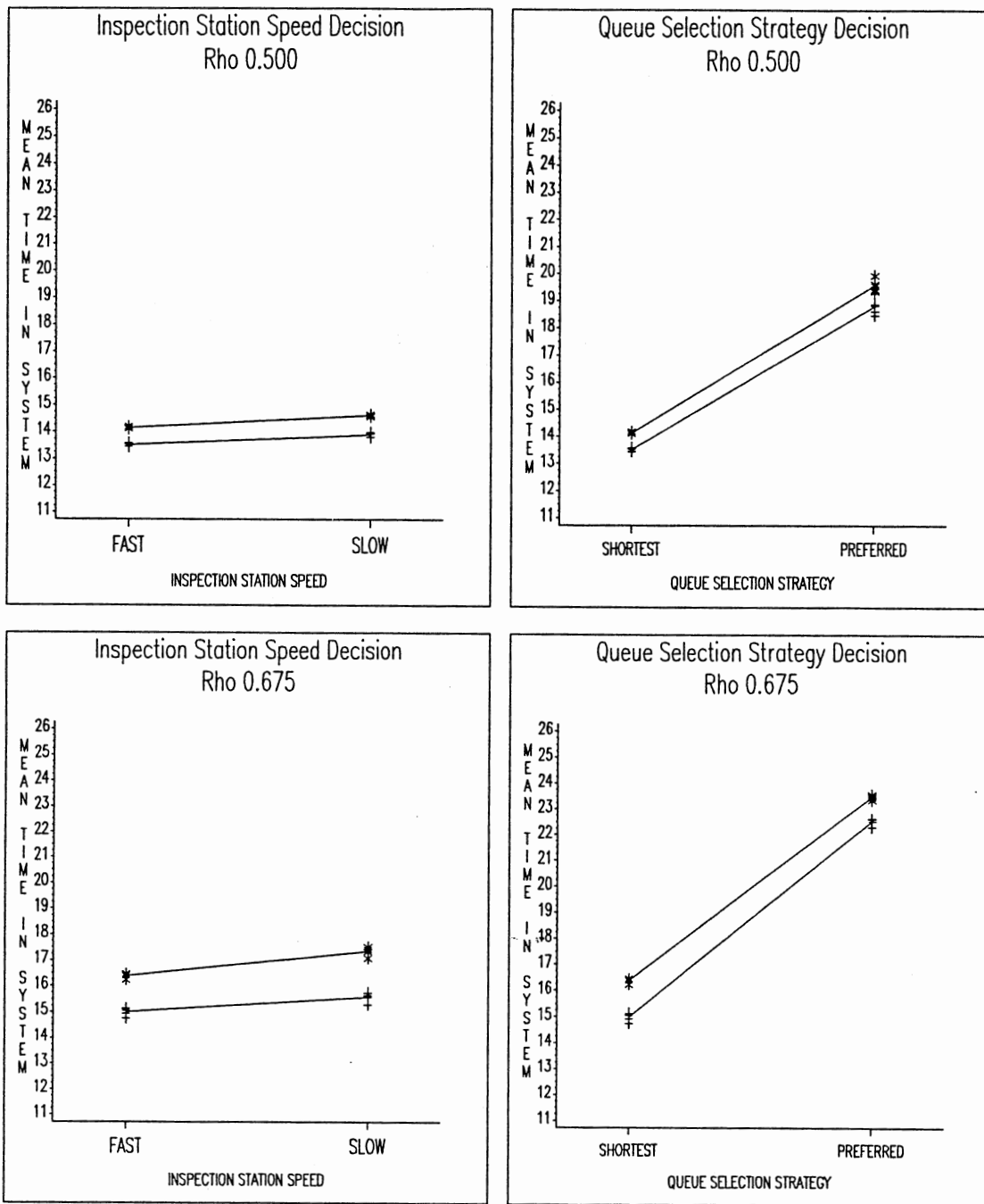
The metamodel with ρ at 0.75, Figure 31, required two iterations of the remedial procedure. These iterations caused the interpolation arrays to ultimately be associated with ρ values of 0.725 and 0.775 rather than 0.60 and 0.80. Even after the iterations, the metamodel CDF curve noticeably lags the base model CDF curve. Even so, the metamodel curve is clearly within limits over its range.

As in the previous scenarios, the PDF curves for both metamodels show the bimodal characteristic. A review of other ρ values (not shown) reveals that this bimodal characteristic remains consistent. The first mode is dominant at lower ρ values while the second mode is dominant at higher ρ values.

Plant Level Validation #1

The four plant level validation #1 graphs for scenario OB3 are shown in Figure 32 on the following page. An inspection of this figure yields the following observations:

- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o the magnitude of the inaccuracies appears to be greater than either of the queueing network cases (QN1 and QN2) but consistent with the first two observation based cases (OB1 and OB2);
- o there is very little discernable inconsistency in the metamodels with ρ at 0.500. With ρ at 0.675, the lines for the inspection speed decision appear to diverge. For the queue selection strategy decision they appear to converge slightly.



+ = Base Model <<<>> * = Meta Model

PLANT OB3

Figure 32. Plant OB3 Validation #1 - Visual Inspection

Plant Level Validation #2

The results for the validation #2 ANOVAs for scenario OB3 are shown in Table XXIII below. As in the prior scenarios, in every case a significant difference in mean time-in-system is detected across the decision alternatives.

TABLE XXIII
ANOVA SUMMARY FOR OB3 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB3/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB3/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB3/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB3/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB3/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB3/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB3/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB3/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject

Plant Level Validation #3

The results of the validation #3 ANOVAs for scenario OB3 are shown in Table XXIV on the following page. As happened for scenario OB2, only in the queue selection strategy decision at rho 0.500 does the table reflect the desired result (no difference in slope) at both α levels. In the all other cases, a difference in slope was detected with a very small OSL (0.0001). As in the previous scenario, these results are not necessarily

intuitively appealing upon a review of Figure 32 since there does not appear to be a marked difference in slopes in any of the graphs.

TABLE XXIV
ANOVA SUMMARY FOR OB3 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB3	Inspection Station Speed	0.500	$H_0: \Delta\text{Slope} = 0$	0.0001	Reject	Reject
OB3	Inspection Station Speed	0.675	$H_0: \Delta\text{Slope} = 0$	0.0001	Reject	Reject
OB3	Queue Selection Strategy	0.500	$H_0: \Delta\text{Slope} = 0$	0.3650	Accept	Accept
OB3	Queue Selection Strategy	0.675	$H_0: \Delta\text{Slope} = 0$	0.0001	Reject	Reject

As before this lack of intuitive appeal is attributed to the difference in variances of the plot points used to estimate the line slopes. As the variance among the plot points at one (or both) ends of the line increases, the confidence interval around the slope estimate widens. This leads directly to an increased probability that the two slopes will not be considered significantly different. In terms of Figure 32, the plot points at the "preferred" end of both lines of the queue selection decision graph with ρ at 0.500 (upper right graph) have a greater variance among them than the points at the "slow" end of the other nearly parallel lines (upper left graph). Specifically, the calculated sample variances for the base and meta points on the upper right graph are 0.3013 and 0.2196 respectively while the same values for the upper left graph are 0.0829 and 0.0378.

The results of the pragmatic validation test #3a are given in Table XXV on the next page. The worst case inconsistency is an acceptable 4.30 percent. It is again interesting that for the queue selection strategy case with ρ at 0.675 the lines are

converging as they were in scenario OB1 and OB2. In every other case in both scenarios, the lines were diverging.

TABLE XXV

ANOVA SUMMARY FOR OB3 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
OB3	Inspection Station Speed	0.500	14.60994	14.44822	1.12
OB3	Inspection Station Speed	0.675	17.34473	16.62901	4.30
OB3	Queue Selection Strategy	0.500	19.57897	19.30435	1.42
OB3	Queue Selection Strategy	0.675	23.49819	2436399	-3.55

Scenario OB4 - Finite Queue Capacity

Introduction

Observation based scenario four is a tree network composed of four machines configured in two stages. The physical configuration and part routings are identical with those of scenario OB1, OB2, and OB3 (see Figure 23). There are three major differences between scenario OB4 and the previous observation based scenarios. The first and most significant difference is that the individual workstations within OB4 have queues with finite capacity. These queues, which can be thought of as WIP buffers, occur on both the input and output side of each workstation.

The queue capacities for the OB4 machines are given in Table XXVI on the next page. As before, the exact parameters were determined based on preliminary experimental runs. The goal of the experimentation was to find queue lengths that would

allow the finite queues to have some impact at low utilization levels without overwhelming the system at high utilization levels. By implementing finite queues in this way, blocking of workstation M1 can occur. Workstation M1 becomes blocked if there is no room in its output queue to hold a processed part. The station remains in a blocked state until room becomes available in this queue. Room becomes available when the currently waiting part can move to the input queue of M2 or directly into processing on M3 or M4.

TABLE XXVI
QUEUE CAPACITIES FOR OB4 MACHINES

MACHINE	INPUT QUEUE CAPACITY	OUTPUT QUEUE CAPACITY
M1	infinite	1
M2	1	infinite
M3	0	infinite
M4	0	infinite

The second change that differentiates scenario OB4 is that the operator assisted setup that was added in scenario OB2 had to be removed. While this breached the intended cumulative nature of the observation based scenarios, the removal was necessary in order for the blocking to have a significant impact. Preliminary runs made with the operator assisted setup in place showed that queueing for the operator paced the progress of parts to the extent that blocking never (or very seldom) occurred at low and moderate stage utilization values.

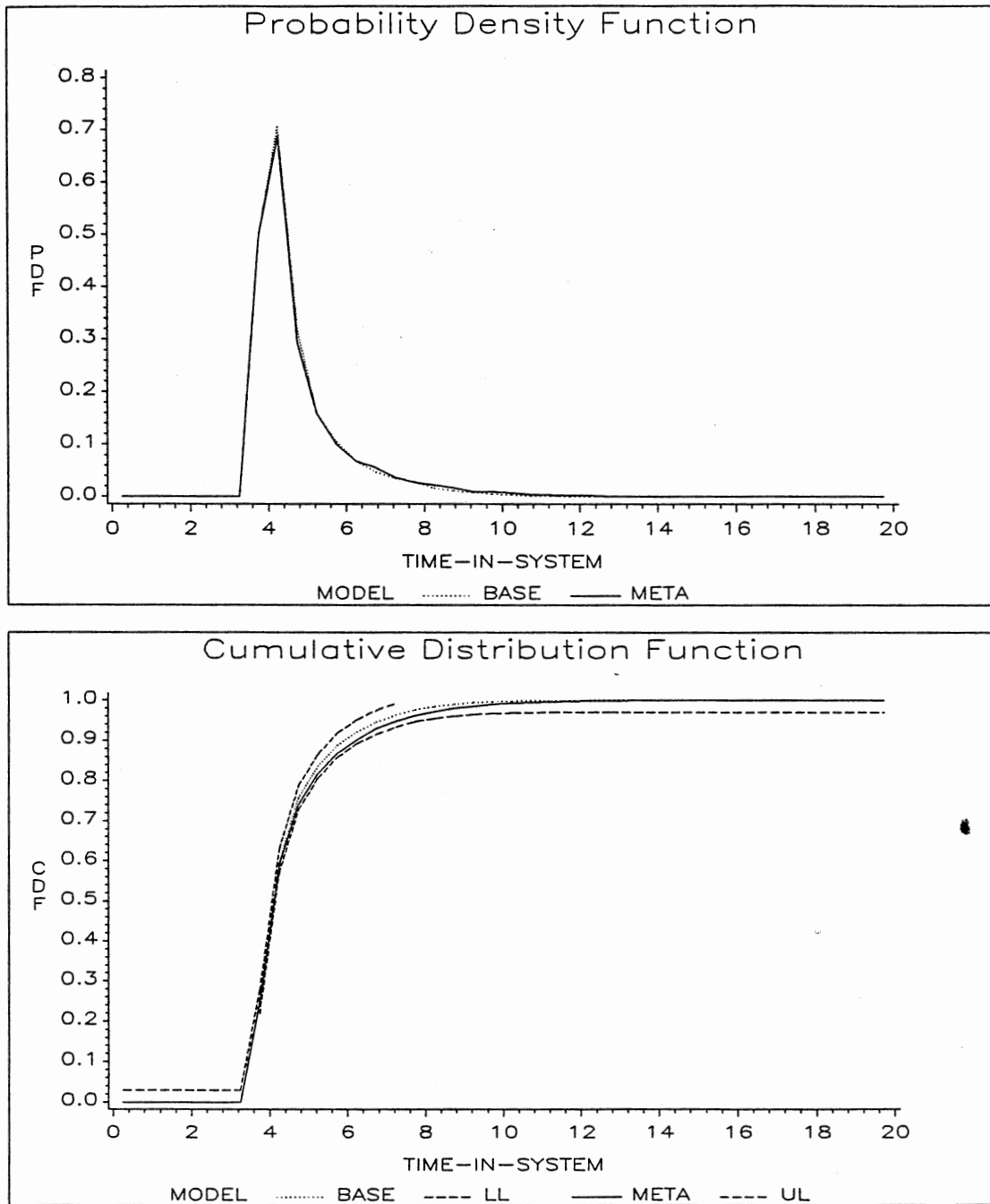
The third major change was required due to the zero input queue capacities in front of machines M3 and M4. When queried these queues would always report that their current queue length was equal to their maximum (i.e., zero). Under this condition, the shortest queue selection rule that considers only queue length (not machine status) would never route a part to the machine. To circumvent this problem the shortest queue selection rule was modified to consider both queue length and machine status.

The service time distributions for M1, M2, M3, and M4 were reinstated to their scenario OB1 values (i.e., prior to setup). The machine failure and operator assisted repair distributions remain as they were in scenario OB3. The arrival process to the workcenter is Poisson, therefore, the interarrival time of parts to the workcenter is exponentially distributed. The mean of the interarrival time distribution is set at 4.00, 2.50, 1.67, or 1.25 depending upon the targeted stage utilization. The corresponding stage utilization values are approximately 0.25, 0.40, 0.60, and 0.80.

The distributional form for the metamodel for this workcenter cannot be determined analytically. The presence of state dependent routings, multiple concurrent resources (for repair only), machine failures, and finite queues violate the product form solution assumptions required for an analytical solution to be calculable.

Workcenter Level Validation

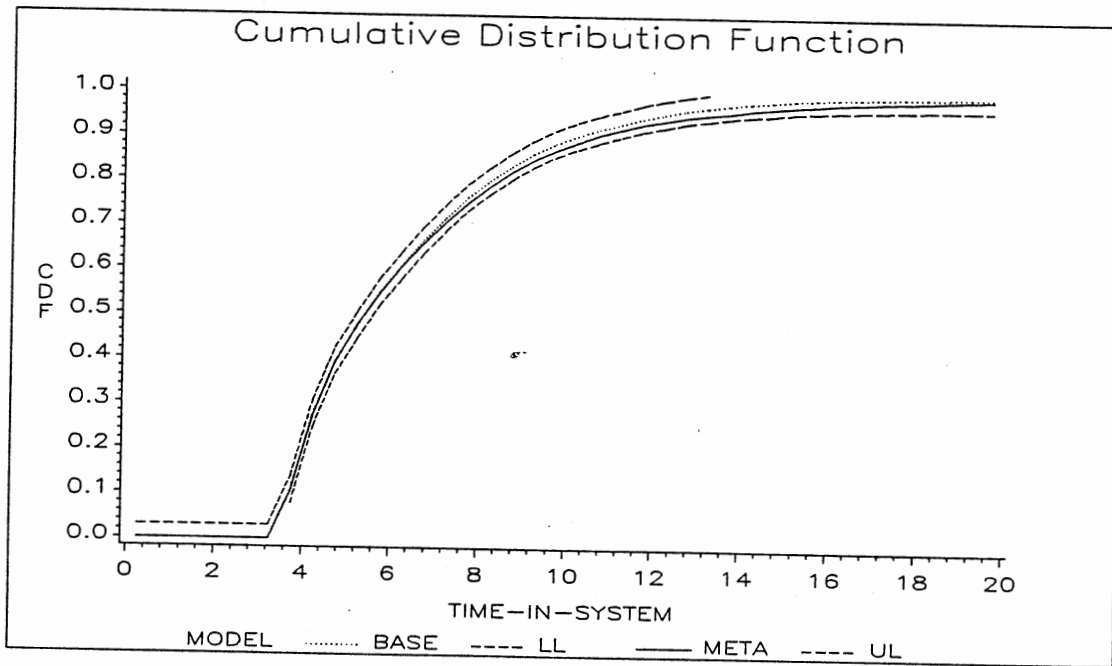
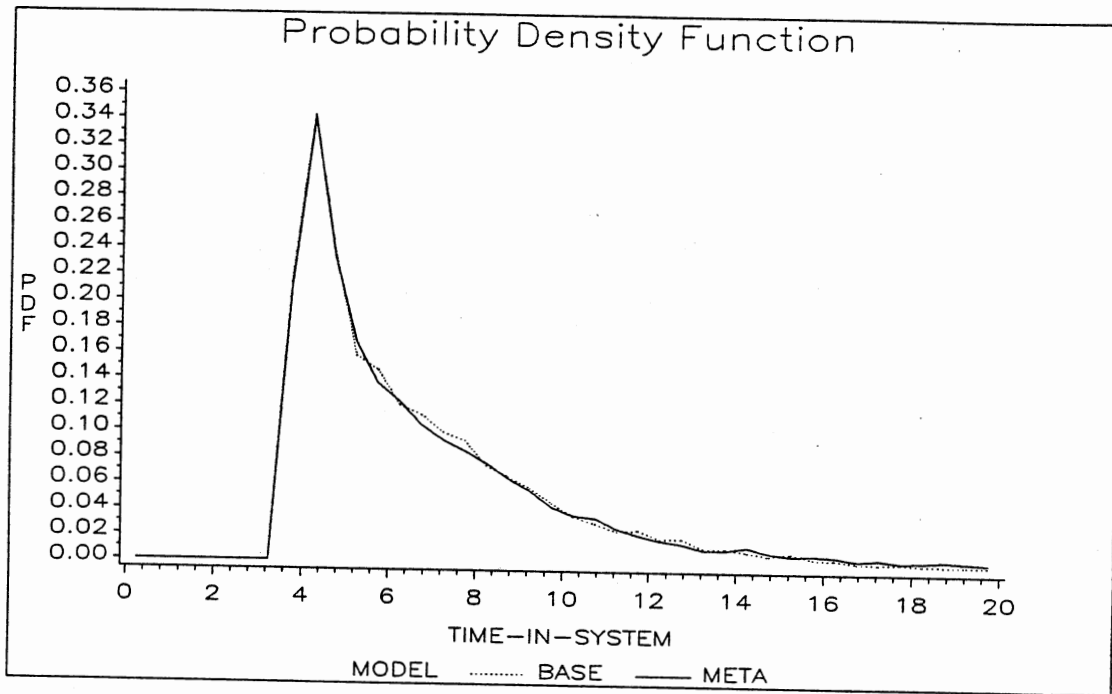
The workcenter validation graphs for scenario OB4 are shown in Figures 33 ($\rho=0.50$) and 34 ($\rho=0.75$) on the following two pages. The metamodel with ρ at 0.50, Figure 33, was produced with no remedial cycles. The metamodel with ρ at 0.75, Figure 34, required two iterations of the remedial procedure. These iterations caused the interpolation arrays to ultimately be associated with ρ values of 0.725 and 0.775 rather than 0.60 and 0.80. Both metamodel CDFs track well through the mode but lag noticeably after the mode. Both metamodels are within the Kolmogorov-Smirnov limits



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB4 - Rho 0.50

Figure 33. Plant OB4 Workcenter Validation - Rho 0.50



LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits

WORKCENTER OB4 - Rho 0.75

Figure 34. Plant OB4 Workcenter Validation - Rho 0.75

over their entire range. Although this is difficult to see on the near vertical part of the CDF curve of Figure 33, the SAS validation program (Appendix G) verifies that no limits were violated.

In contrast to the previous observation based scenarios, the PDF curves for the metamodels do not exhibit the bimodal characteristic. A review of other ρ values (not shown) reveals that this unimodal characteristic remains consistent. It is the author's hypothesis that the modification of the queue selection rule to include consideration of machine status induced this change.

Within the discussion of results of Scenario OB1, it was hypothesized that the first mode of the bimodal PDF was representative of time-in-system for parts that were queued to a stage two machine that was idle. The second mode was representative of parts queued to a busy machine. The valley between the modes is created by the queue selection rule that, under some circumstances, prefers a busy machine over an idle one. This misguided preference occurs when a low numbered machine (e.g., M1) has no queue but is busy while a higher numbered machine (e.g., M2) has no queue and is idle. Under these circumstances, the queue selection algorithm would "see" a tie between the competing machines since both have a waiting queue length of zero. The algorithm breaks ties in favor of the lower numbered machine. Thus, in the example, the part would be queued to M1 although it was busy.

Under the current scenario, the valley between the modes is lost. It is the author's hypothesis that this results from a modification of the queue selection rule to consider machine status. The net result is that a smaller proportion of parts are queued to busy machines since situations similar to the example above would never be seen as a tie. The part would have been definitively queued to M2, the idle machine. Pictorially, this can be thought of as "shaving" part of the peak of the second mode to fill the valley.

Plant Level Validation #1

The four plant level validation #1 graphs for scenario OB4 are shown in Figure 35 on the next page. An inspection of this figure yields the following observations:

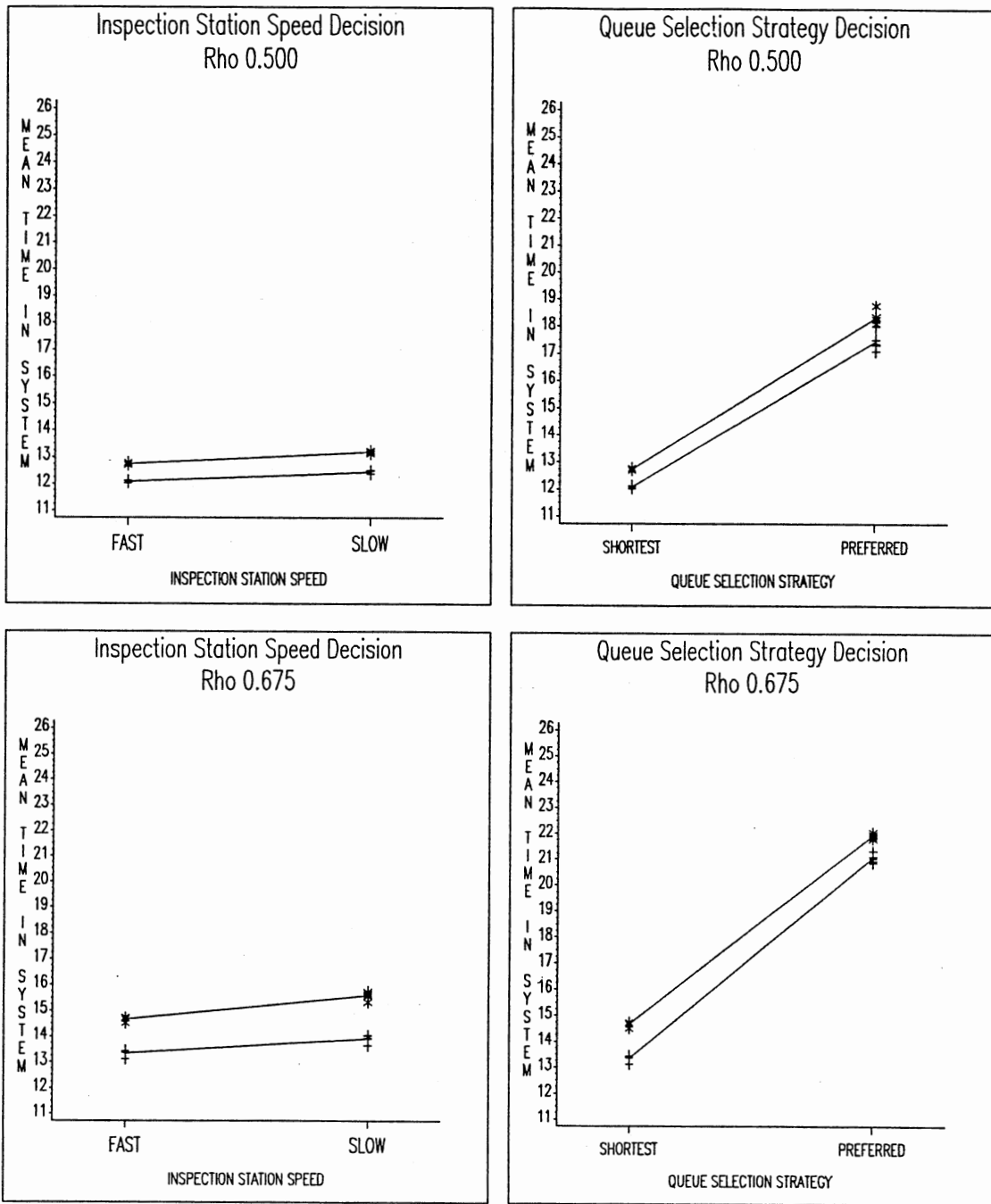
- o the metamodels are approximately accurate and approximately consistent;
- o all metamodels are inaccurate on the high side;
- o the magnitude of the inaccuracies appears to be greater than either of the queueing network cases (QN1 and QN2) but consistent with the first three observation based cases (OB1, OB2, and OB3);
- o there is very little discernable inconsistency in the metamodels with ρ at 0.500. With ρ at 0.675, the lines for the inspection speed decision appear to diverge. For the queue selection strategy decision they appear to converge slightly.

Plant Level Validation #2

The results for the validation #2 ANOVAs for scenario OB4 are shown in Table XXVII on page 143. As in the prior scenarios, in every case a significant difference in mean time-in-system is detected across the decision alternatives.

Plant Level Validation #3

The results of the validation #3 ANOVAs for scenario OB4 are shown in Table XXVIII on page 143. As was true for scenario OB2, only in the queue selection strategy decision at ρ 0.500 does the table reflect the desired result (no difference in slope) at both α levels. In the all other cases, a difference in slope was detected with a small OSL (0.0012). Again, these results are not necessarily intuitively appealing upon a review of Figure 35 since there does not appear to be a marked difference in slopes in any of the graphs.



+ = Base Model <<<>>> * = Meta Model

PLANT OB4

Figure 35. Plant OB4 Validation #1 - Visual Inspection

TABLE XXVII
ANOVA SUMMARY FOR OB4 PLANT LEVEL VALIDATION #2

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB4/Base	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB4/Meta	Inspection Station Speed	0.500	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB4/Base	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB4/Meta	Inspection Station Speed	0.675	$H_0: \mu_F = \mu_S$	0.0001	Reject	Reject
OB4/Base	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB4/Meta	Queue Selection Strategy	0.500	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB4/Base	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject
OB4/Meta	Queue Selection Strategy	0.675	$H_0: \mu_P = \mu_S$	0.0001	Reject	Reject

TABLE XXVIII
ANOVA SUMMARY FOR OB4 PLANT LEVEL VALIDATION #3

MODEL	TREATMENT	RHO	HYPOTHESIS	OSL	$\alpha = 0.05$	$\alpha = 0.01$
OB4	Inspection Station Speed	0.500	$H_0: \Delta Slope = 0$	0.0001	Reject	Reject
OB4	Inspection Station Speed	0.675	$H_0: \Delta Slope = 0$	0.0001	Reject	Reject
OB4	Queue Selection Strategy	0.500	$H_0: \Delta Slope = 0$	0.2957	Accept	Accept
OB4	Queue Selection Strategy	0.675	$H_0: \Delta Slope = 0$	0.0012	Reject	Reject

As before this lack of intuitive appeal is attributed to the difference in variances of the plot points used to estimate the line slopes. As the variance among the plot points at one (or both) ends of the line increases, the confidence interval around the slope estimate widens. This leads directly to an increased probability that the two slopes will not be considered significantly different. In terms of Figure 35, the plot points at the

"preferred" end of both lines of the queue selection decision graph with ρ at 0.500 (upper right graph) have a greater variance among them than the points at the "slow" end of the other nearly parallel lines (upper left graph). Specifically, the calculated sample variances for the base and meta points on the upper right graph are 0.3409 and 0.2648 respectively while the same values for the upper left graph are 0.0626 and 0.0520.

In this scenario there is also one decision case with an OSL that is in the "reject" range but is slightly higher than it was in previous scenarios (0.0012). An inspection of Figure 35 reveals that the plot point variance appears slightly greater for the "preferred" end of the base model line of the queue selection decision graph with ρ at 0.675 (lower right graph). The calculated sample variance for these points is 0.1808. This value is bracketed by the lowest "accept" values (variance of 0.2648 with an OSL of 0.2957) and highest "reject" values (variance of 0.0626 with an OSL of 0.0001) in the preceding paragraph. Since these results are consistent with the plot point variance hypothesis, the hypothesis remains credible.

The results of the pragmatic validation test #3a are given in Table XXIX on the following page. The worst case inconsistency is an acceptable 4.68 percent. It is again interesting that for the queue selection strategy case with ρ at 0.675 the lines are converging as they were in scenario OB1, OB2, OB3. In every other case in each scenario, the lines were diverging.

Inter-Scenario Comparisons

The focus of the analysis in the preceding sections has been primarily intra-scenario. It is appropriate to note some observations regarding inter-scenario results. The observations below result from the simultaneous consideration of the results of plant level validation #3a for each scenario (Tables XIII, XVI, XIX, XXII, XXV, and XXIX).

TABLE XXIX

ANOVA SUMMARY FOR OB4 PLANT LEVEL VALIDATION #3a

MODEL	TREATMENT	RHO	META	CORRECTED META	PERCENT ERROR
OB4	Inspection Station Speed	0.500	13.19580	13.00810	1.44
OB4	Inspection Station Speed	0.675	15.58739	14.89030	4.68
OB4	Queue Selection Strategy	0.500	18.32305	17.92185	2.24
OB4	Queue Selection Strategy	0.675	21.92960	22.84682	-4.01

- o In all eight pairs of the observation-based scenario decision cases, the absolute value of the percent error for the $\rho = 0.50$ case is less than that for the corresponding $\rho = 0.675$ case. This observation is consistent with the behavior of the metamodels with respect to the remedial procedure. The higher ρ , the more difficult it is for the metamodel to track the base model within Kolmogorov-Smirnov limits. It is an interesting contrast to note that while metamodels encounter difficulty at high ρ values, queueing approximations (see Chapter III) perform well. This suggests potential future research in the area of combined metamodels and queueing approximations.
- o In three out of four of the pairs of queueing network scenario decision cases, the absolute value of the percent error for the $\rho = 0.50$ case is greater than that for the corresponding $\rho = 0.675$ case. This result is counter intuitive and perhaps warrants additional research.
- o Out of the sixteen individual observation-based decision cases, twelve had divergent base and metamodel lines and four had convergent lines. In each case the convergent lines occurred with the queue selection strategy decision at $\rho = 0.675$. This would suggest that the type of decision and ρ value have a

significant bearing on sign of the inconsistency error. Further research is suggested to explore this relationship.

- o For Scenarios OB2, OB3, and OB4, the inconsistency errors across decisions but within ρ values are comparable. The inconsistency errors across ρ values but within decisions are less comparable. This suggests that the metamodels considered in this research (particularly the complex scenarios) are more sensitive to the value of ρ than they are to the type of decision.
- o The stability of the level of inconsistency error across scenarios but within decision cases and ρ value, suggests that calibration and/or correction factors could be incorporated into a metamodel to improve its consistency. Further research is needed to learn how such factors could be included in the metamodel development and validation procedures.

The observations presented in this section suggest many things about metamodels and the relationship between model complexity, decision cases, and ρ values. Additional research is needed to substantiate these observations across a wider range of scenarios. Ultimately this research could lead to insights into fundamental relationships as discussed in Chapter I.

Computer Execution Times

The primary goal for this research is to develop a methodology for hybrid metamodeling of complex manufacturing systems and assess its viability. This goal is motivated by an interest in reducing the time required for simulation model execution. Given this goal and motivation, it may seem curious that no research objective was developed to specifically measure the time savings achieved under the various scenarios. The reason for this is twofold. First, the dominant focus of this effort is on a practicable

methodology not on the ultimate time savings achievable through its use. The focus is on the question "does it work?" rather than the question "how well does it work?"

The second reason that measurement of execution times is not a formal objective is the intuitive appeal of the speed of the metamodels. This appeal stems from the fact that metamodels are inherently "less complex" than their base model counterparts. Complexity is reduced in at least two dimensions: (1) only one random variate must be generated for the metamodel whereas a base model may require many, and (2) no "internal" logic must be exercised for the metamodel whereas a base model may require embedded logic for queue selection rules, machine breakdown and repair, etc.

While not a formal objective it is still prudent to examine the impact of metamodels on execution time within this research. To accomplish this, average computer execution time savings for each scenario is calculated and shown in Table XXX. These results reflect computer time savings ranging from approximately fifteen percent for the least complex models to approximately twenty percent for the most complex models.

TABLE XXX
SUMMARY OF COMPUTER EXECUTION TIME SAVINGS

SCENARIO	PERCENT TIME REDUCTION FOR METAMODELS
QN1	15.3 %
QN2	14.6 %
OB1	15.0 %
OB2	18.8 %
OB3	18.4 %
OB4	20.2 %

Two factors need to be considered in putting these time savings into perspective relative to the potential of metamodeling. First, the scenarios considered in this research are relatively simple. They all are composed of four or fewer machines and only one part type. If more complex scenarios were evaluated it would be reasonable to expect the percentage time savings to increase. Second, within the OOM environment, the metamodel sampling procedure is written in a straightforward but not necessarily efficient form. The sampling procedure includes a left-to-right interval search of the empirical grouped distribution function (see Appendix E for details). This search could be performed in a more efficient, application specific manner to enhance the time savings. This would be particularly valuable for empirical grouped distribution functions containing many cells.

Summary Of Experimental Results

The methodologies and results presented in this chapter complete the fulfillment of research Objectives 1 through 5. Objective 1 is the development of a metamodel selection procedure. The metamodel selection procedure presented in Chapter VII is implemented as described under the Metamodel Usage sub-heading of Scenario OB1. This fulfills Objective 1.

Objective 2 is the development of the observation based metamodel procedure. It is fulfilled through the triad of algorithms described under the sub-headings Metamodel Development, Metamodel Validation, and Metamodel Remedial Procedure of Scenario OB1.

The development of the queueing network metamodel procedure is Objective 3. This objective was included to demonstrate the robustness of the metamodeling concept. Queueing network metamodels are case-specific. Each queueing network model must be individually solved to yield the metamodel cumulative distribution function based on its

unique characteristics. The introductory sections of Scenarios QN1 and QN2 are exemplary of this procedure and thus fulfill the Objective 3.

Objective 4 is the demonstration of proof-of-concept through prototype implementation. The OOM environment used for implementation is described in Chapter VI and Appendix A. The proof of concept is demonstrated via the implementation of the results of the first three objectives within the OOM environment. The workcenter level validations described within each scenario provide evidence of the viability of the developed procedures.

The validation of metamodels through consistency of plant level decisions is Objective 5. The results presented in this chapter demonstrate the pragmatic acceptability of the developed metamodeling methodology. The outcomes of the statistical tests were less desirable but are mitigated by the impact of the large sample size associated with the experimental design.

CHAPTER IX

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Introduction

This chapter presents concluding thoughts about this research effort. It includes a summary of the research in light of the proposed objectives, contributions to the body of knowledge attributable to the research, and recommendations for extended and/or additional research in the area.

Research Summary

The goal of this research was to develop a methodology for hybrid metamodeling of manufacturing systems within a simulation framework. The goal was systematically addressed through six research objectives. The sections below review the accomplishments in each of these areas.

Metamodel Selection Procedure

The first objective was the development of a procedure to determine which workcenters within a multi-workcenter simulation model are candidates for a metamodel. This procedure is the subject of Chapter IV. The derived procedure involves two major steps: the determination of candidate workcenters and the determination of the availability of a valid metamodel. Candidate workcenters are those whose desired performance measures within the experimental frame do not conflict with the metamodel

aggregate behavior. Table IX on page 84 summarizes the results of this process within the following restrictions:

- o Potential performance measures are given by Law [1986].
- o Metamodel aggregate behavior preserves the time-in-system distribution.
- o the OSU OOM environment is the simulator.¹

The second step of the procedure is the determination of an available valid metamodel. Within this research, this determination is made within the modeling environment through two checks. The first check determines whether a metamodel has been previously loaded for the workcenter. The second check determines whether the desired value of the input parameter (mean arrival rate divided by mean stage service rate) is within the bounds of the data used to create the metamodel. If both checks are successfully passed then a metamodel is available and can be implemented.

Two caveats apply to the procedure outlined above. First, a metamodel is valid only for the physical configuration and operational characteristics for which it was developed. If either of these is modified in any way then the metamodel must be re-validated before use. Second, the existence of a metamodel that has been validated for a set of ρ values, does not guarantee that it is valid (within the defined Kolmogorov-Smirnov limits) for every possible ρ value within its allowed range.

Within this research effort each metamodel was validated at three ρ values, 0.50, 0.675, and 0.75 using the validation procedure presented in Chapter VIII. In pursuing these validations, the following judgmental guidelines emerged for assessing the validity of a metamodel at other ρ values:

- o if the desired ρ value is greater than 0.25 and less than 0.60, a metamodel interpolation range less than or equal to 0.20 will produce a valid metamodel;

¹The metamodeling concept is valid in many potential environments. This restriction exists because the OSU OOM environment cannot currently be used for all potential performance measures.

- o if the desired ρ value is greater than 0.60 and less than 0.70, a metamodel interpolation range less than or equal to 0.10 will produce a valid metamodel;
- o if the desired ρ value is greater than 0.70 and less than 0.80, a metamodel interpolation range less than or equal to 0.05 will produce a valid metamodel;
- o in general, as the ρ value increases the interpolation range should decrease;
- o in general, for a given ρ , as the complexity of the workcenter structure and/or operating characteristics increases, the interpolation range for a metamodel should be narrowed;

Using the procedures and guidelines summarized above, the objective of developing a metamodel selection procedure was accomplished.

Observation-Based Metamodel Procedure

The second objective was the creation of a procedure to develop and validate (at the workcenter level) an observation-based metamodel. In the context of this research, an observation-based metamodel is a metamodel developed by observing the behavior of a detailed (base model) version of the workcenter. This procedure is required when no closed form solution is available for the time-in-system distribution of the base model. The development and validation procedures were both presented in detail within the Scenario OB1 Section of Chapter VIII.

One of the most significant disappointments of this research was that no curve fitting based method was discovered for observation-based metamodel development. Several early ad hoc attempts using stepwise multiple regression failed to provide solutions that were competitive with the interpolation based procedure. In defense of the interpolation procedure, it is attractive because it is conceptually and graphically easy to comprehend and because none of the metamodels within this research required more than two cycles of the remedial procedure to converge.

Using the procedures summarized above, the objective of creating an observation-based metamodel development procedure was accomplished.

Queueing Network Metamodel Procedure

The third objective was the creation of a procedure to develop a queueing network metamodel. In the context of this research, a queueing network metamodel is a metamodel developed by solving for a closed form solution of the time-in-system distribution for the workcenter. This procedure can be used when a workcenter meets the assumptions required for a product form solution with non-overtaking paths. The procedure for developing this type of metamodel is discussed within the Scenario QN1 and Scenario QN2 Sections of Chapter VIII. Using the procedure presented in these two sections, the objective of creating a queueing network metamodel development procedure was accomplished.

Proof of Concept Via Prototype Development

The fourth objective was the implementation of the results of objectives one through three in a way that demonstrates proof of concept. The implementation was accomplished within the OSU OOM environment presented in Chapter VI. The structure of the user interaction with the OOM environment and the proof of concept results are presented in Chapters VII and VIII. The reduction in computer run times produced by implementing metamodels (15% to 20%) is documented in Table XXX on page 147.

The fourth objective as presented in the July, 1991 Research Proposal and Chapter IV includes the development of a procedure for on-line parameterization of the metamodel. This procedure was intended to accommodate situations in which the metamodel parameter was either unknown prior to initiating the simulation run or dynamically changing within the simulation run. As discussed in Chapter VIII this requirement eventually became moot for two reasons. First, due to the nature of the

cases used within this research, the metamodel parameter can be determined from flow balance analysis prior to the run, thus eliminating the unknown parameter case. Second, the metamodels in this research are not time-varying models. By their very nature they are suitable only for steady state not transient modeling, thus eliminating the dynamic case.

With the above exception noted, the results of Chapters VI, VII, and VIII provide evidence that the objective of implementation and proof of concept was accomplished.

Plant Level Validation

The fifth objective was the evaluation of the methodology and metamodels at the plant level through simulation experiments. This evaluation was accomplished through the plant level validation results presented in Chapter VIII. The validation was judged based on both empirical and statistical analysis of the consistency between a decision made using a base model and the same decision made using a corresponding metamodel.

The results presented were satisfying from the standpoint of visual empirical evidence and pragmatic error analysis. The results were less satisfying from a pure statistical analysis point of view. Arguments were presented that discount the statistical analysis in favor of the empirical and pragmatic results due to the impact of the large sample sizes involved.

With the above subjective argument noted, the results presented in Chapter VIII constitute the accomplishment of metamodel evaluation and validation at the plant level.

Future Research

The final objective was the identification of further research in this area. Many areas of potential research exist within the broad context of hybrid metamodeling. Published results to date in this area barely scratch the surface of what might be

accomplished via approximate modeling techniques. The final section of this chapter presents ideas on further research that grew out of this effort.

Research Contributions

Published results in the area of hybrid metamodeling are scant. One of the major intended contributions of this research was to stimulate additional research in this area through a demonstration of the viability of the concept. The vehicle used to demonstrate proof of concept was the bottom-up hierarchical object-oriented manufacturing simulation environment currently under development at OSU's Center for Computer Integrated Manufacturing.

The completion of the research objectives as documented in the previous section makes the following contributions to the area of advanced simulation modeling within Industrial Engineering:

- o a metamodeling methodology has been developed that appears to have broad application to hierarchical simulation models of manufacturing systems;
- o the viability of the methodology, in terms of decision consistency, has been demonstrated over a narrowly defined set of scenarios;
- o a reduction of computer processing time through metamodeling has been demonstrated, thereby, making on-line simulation and search-driven simulation more feasible;
- o an implementation procedure for plug-compatible metamodels has been shown effective within the OSU OOM environment;
- o the need for additional research aimed at extending and/or generalizing the results of this effort such that they become available to mainstream practitioners has been stimulated;
- o the effectiveness of the OSU OOM environment as a research test bed has been demonstrated.

Recommendations For Future Research

As a result of the research conducted in this study, recommendations of additional research can be made. They are described in the sections that follow.

Metamodel Parameterization

All the metamodels in this research were parameterized on ρ , the expected stage utilization of the workcenter. In each case the workcenter structure and operating characteristics were held constant. Controlled experimentation is needed to determine if workcenter structure and/or operating characteristics are viable metamodel parameters under constant utilization.

Controlled experimentation is also needed to determine if time-parameterized metamodels are feasible. Successful implementation of this concept would allow metamodels to be used for transient analysis as well as steady-state analysis. This research could also lead to investigation of dynamic substitution of metamodels. When simulation conditions warrant, an appropriate metamodel would be "plugged" into the overall model until the conditions changed sufficiently to call for another substitution. An even broader interpretation of a dynamic metamodel is a "learning" metamodel. Such a metamodel could conceivably use neural nets as a learning vehicle to change itself over the course of time as conditions warrant.

Metamodel Accuracy

The primary validation measure for the metamodels in this research as prescribed by Objective 5 is consistency of decisions. Figure 19 defines this error and differentiates it from an error of inaccuracy. The plant level validation #1 results presented in Figures 20, 22, 26, 29, 32, and 35 clearly show that varying levels of inaccuracy are present in the metamodels. Further research is suggested to investigate creating more accurate

metamodels, perhaps through the use of multiple variables (e.g., $E[\rho]$ and $\text{Var}[\rho]$) rather than a single variable (e.g., $E[\rho]$) to parameterize the metamodel.

Metamodel Calibration

Several areas of further research were noted in the inter-scenario section of Chapter VIII. These areas were primarily based on the consistent behavior of metamodels across the scenarios. The observations suggest that calibration of metamodels through additive and/or multiplicative correction factors might be used to improve both inaccuracy and inconsistency. Additional research is needed to substantiate these observations and develop a methodology.

Metamodel Interpolation Ranges

Guidelines for acceptable interpolation ranges for metamodels were presented above. These guidelines were developed based upon the narrow set of scenarios used in this research. Research is needed to enhance and validate these guidelines on a significantly broader set of scenarios.

Metamodel Analytic Forms

One of the original intentions of this research was to discover analytic forms for metamodels or classes of metamodels. No analytic forms were found in this study that were competitive with the interpolation algorithms in terms of accuracy or efficiency. A more rigorous effort is needed in this area to continue to pursue "insight not numbers."

Metamodel Target Behavior

The target behavior to be approximately maintained by all of the metamodels within this research was time-in-system for a single part type. Research is needed to investigate the viability of alternative behaviors (e.g., maximum WIP, time-in-queue,

etc.) and/or multiple behaviors (e.g., time-in-system for multiple part types) for metamodels.

Multiple and Nested Metamodels

The validation activities within this research were limited to cases where a single metamodel existed within a plant scenario. Research is needed to determine whether acceptable levels of consistency can be maintained if a plant model contains two or more metamodels. Research is also needed to determine if acceptable consistency can be achieved with two tiered metamodels (i.e., a group of two or more metamodels can be metamodeled with some form of "super" metamodel). This type of two-tiered metamodel would be useful if, for instance, all the workcenters within a department of a multi-department model were amenable to metamodeling. For either multiple or nested metamodels, research is needed bound the potential resource savings (i.e., computer execution time savings) to be obtained by cumulative applications of metamodeling.

Justification of Metamodeling

Within this research, the use of metamodels is rationalized based primarily on a reduction in amount of computer processing time required. These savings are particularly relevant when a metamodel is used repeatedly over time or when the reduction enables the use of simulation in real-time or search based applications. Additional research is needed to quantify these potential savings and to develop a methodology for determining the conditions under which the creation and maintenance of a metamodel is cost justified.

Tangential Research

The additional research areas identified above are either extensions of this research or otherwise directly related to the stated objectives. Other areas of research were

discovered in the course of this effort that are tangentially related. A brief listing of these areas is as follows:

- o The ANOVAs prepared for plant level validations #2 and #3 both indicated that random number sets were a significant source of variation. Ideally, the choice of random number seed should not be a significant source of variation. This result emphasizes the need for continued efforts in this already active area of research.
- o Investigation during the simulation run design phase of this research (see Appendix B) indicated that multiple stream random number generation was superior to single stream generation. Additional research is needed to fully investigate the ramifications of this effect and understand the rationale behind it (see also [Mize 1973]).

BIBLIOGRAPHY

- Agrawal, S.C. 1985. Metamodeling: A Study of Approximations in Queueing Models. Research Notes and Reports, Computer Systems Series. Cambridge, MA: The MIT Press.
- Ahrens, J.H. and U. Dieter. 1974. "Computer Methods For Sampling From The Exponential and Normal Distributions." Computing 12: 223-246.
- Alla, H., P Ladet, J. Martinez, and M. Silva-Suarez. 1985. "Modeling and Validation of Complex Systems by Colored Petri Nets: Application to Flexible Manufacturing System." In Advances in Petri Nets 1984, edited by G. Goos and J. Hartmanis: 15-31. Springer Verlag: LNCS 188.
- Balbo, G., G. Chiola, and G. Franceschinis. 1989. "Stochastic Petri Net Simulation for the Evaluation of Flexible Manufacturing Systems." In Proceedings of the 1989 European Simulation Multiconference: 5-12.
- Baskett, F., K.M. Chandy, R.R. Muntz, and F.G. Palacios. 1975. "Open, Closed, and Mixed Networks of Queues With Different Classes of Customers." Journal of the ACM 22: 248-260.
- Basnet, C.B. 1991. "On-Line Scheduling and Control of Random Flexible Manufacturing Systems Within an Object Oriented Framework." Ph.D. dissertation, Oklahoma State University.
- Basnet, C., H. Bhuskute, P. Farrington, M. Kamath, J. Mize, and D. Pratt,. 1990. "OSU-CIM Library of Objects for Modeling and Simulation of Discrete Part Manufacturing Systems." Center for Computer Integrated Manufacturing, Working Paper Series: CIM-WPS-90-CB1. Oklahoma State University.
- Basnet, C., P. Farrington, D. Pratt, M. Kamath, C. Karacal, and J. Mize. 1990. "Experiences in Developing an Object-Oriented Modeling Environment for Manufacturing Systems." In Proceedings Of The 1990 Winter Simulation Conference, edited by O. Balci, R.P. Sadowski, and R.E. Nance, 477-481. Piscataway, NJ: IEEE.
- Beaumariage, T.G. 1990. "Investigation of an Object Oriented Modeling Environment for Manufacturing Systems." Ph.D. dissertation, Oklahoma State University.

- Bitran, G.R. and D. Tirupati. 1988. "Multiproduct Queueing Networks with Deterministic Routing: Decomposition Approach and the Notion of Interference." Management Science 34, no. 1: 75-100.
- Blanning, R.W. 1975. "The Construction and Implementation of Metamodels." Simulation, June 1975: 177-184.
- Boxma, O.J., F.P. Kelly, and A.G. Konheim. 1984. "The Product Form for Sojourn Time Distributions of Cyclic Exponential Queues." Journal of the Association of Computing Machinery 31, no. 1: 128-133.
- Buzacott, J.A. and J.G. Shanthikumar. 1980. "Models for Understanding Flexible Manufacturing Systems." AIIE Transactions 12: 339-350.
- Buzacott, J.A. and D.D. Yao. 1986. "Flexible Manufacturing Systems: A Review of Analytical Models." Management Science 32, no. 7: 890-905.
- Chandy, K.M., U. Herzog, and L. Woo. 1975. "Parametric Analysis of Queueing Networks." IBM Journal of Research and Development 19: 36-42.
- Chen, L. and C. Chen. 1990. "A Fast Simulation Approach For Tandem Queueing Systems." In Proceedings Of The 1990 Winter Simulation Conference, edited by O. Balci, R.P. Sadowski, and R.E. Nance, 539-546. Piscataway, NJ: IEEE.
- Cheng, R.C.H. 1977. "The Generation Of Gamma Variables With Non-Integral Shape Parameter." Applied Statistics 26: 71-75.
- Chiu, W.W. and W.M. Chow. 1978. "A Performance Model of MVS." IBM Systems Journal 17: 444-462.
- Conway, A.E. and N.D. Georganas. 1989. Queueing Networks - Exact Computational Algorithms. Cambridge, Massachusetts: The MIT Press.
- Cook, T.M. and R.A. Russell. 1976. "A Survey of Industrial OR/MS Activities in the 70's." In Proceeding of the 8th Annual Conference of the American Institute of Decision Sciences.
- Dietrich, B.L. and B.M. March. 1985. "An Application of a Hybrid Approach to Modeling A Flexible Manufacturing System." Annals of Operations Research 3: 263-276.
- Daduna, H. 1982. "Passage Times for Overtake-Free Paths in Gordon-Newell Networks." Advanced Applied Probability 14: 672-686.

- Endesfelder, T. and H. Tempelmeier. 1987. "The SIMAN Module Processor - A Flexible Software Tool for the Generation of SIMAN Simulation Models." In Simulation in Computer Integrated Manufacturing and Artificial Intelligence Techniques, edited by J. Retti, 38-43. San Diego, CA: The Society of Computer Simulation.
- Friedman, L.W. 1984. "Establishing Functional Relationships in Multiple Response Simulation: The Multivariate General Linear Metamodel." In Proceedings of the 1984 Winter Simulation Conference. Dallas, TX: 285-289.
- Friedman, L.W. 1986. "Exploring Relationships in Multiple-Response Simulation Experiments." Omega 14, no. 6: 498-501.
- Geoffrion, A. 1976. "The Purpose of Mathematical Programming is Insight, Not Numbers." Interfaces 7, no. 1: 81-92.
- Gelenbe, E. and G. Pujolle. 1987. Introduction to Queueing Networks. New York, NY: John Wiley and Sons.
- Goldberg, A. and D. Robson. 1989. SMALLTALK-80: The Language and Its Implementation. Reading, MA: Addison-Wesley.
- Gordon, W.J. and G.F. Newell. 1967. "Closed Queueing Systems with Exponential Servers." Operations Research 15: 254-265.
- Haider, S.W., D.G. Noller, and T.B. Robey. 1986. "Experiences With Analytic and Simulation Modeling for a Factory of the Future Project at IBM." In Proceedings of the 1986 Winter Simulation Conference, edited by J. Wilson, J. Henriksen, and S. Roberts, 641-647.
- Hamming, R.W. 1962. Numerical Methods For Scientists and Engineers. McGraw-Hill, Inc.
- Hicks, C.R. 1964. Fundamental Concepts In The Design of Experiments. New York, NY: Holt, Rinehart and Winston.
- Holliday, M.A. and M.K. Vernon. 1987. "A Generalized Timed Petri Net Model for Performance Analysis." In Transactions on Software Engineering 13: 1297-1310. IEEE.
- Ignall, E.J. and P. Kolesar. 1978. "Using Simulation to Develop and Validate Analytical Models: Some Case Studies." Operations Research 26, no. 2: 237-253.

- Ignall, E.J. and P. Kolesar. 1979. "On Using Simulation To Extend OR/MS Theory: The Symbiosis Of Simulation And Analysis." Chap. in Current Issues in Computer Simulation, ed. N.R. Adam and A. Dogramaci, 223-233. New York, NY: Academic Press.
- Jackson, J.R. 1957. "Networks of Waiting Lines." Operations Research 5: 518-521.
- Jackson, J.R. 1963. "Jobshop-like Queueing Systems." Management Science 10: 131-142.
- Kamath, M. 1989. "Analytical Performance Models for Automatic Assembly Systems." Ph.D. dissertation, University of Wisconsin-Madison.
- Kamath, M. 1991. "A Two-moment Model for the Approximate Analysis of General Tandem Queues with Blocking." Center for Computer Integrated Manufacturing, Working Paper Series: CIM-WPS-91-MK2. Oklahoma State University.
- Kamath, M., H. Bhuskute, and M. Duse. 1991. "Fast Simulation Techniques for Queueing Networks." Center for Computer Integrated Manufacturing, Working Paper Series: CIM-WPS-91-MK1. Oklahoma State University.
- Kamath, M. and J.L. Sanders. 1987. "Analytical Methods for Performance Evaluation of Large Asynchronous Automatic Assembly Systems." Large Scale Systems 12, no. 2: 143-154.
- Kamath, M. and J.L. Sanders. 1991. "Modeling Operator/Workstation Interference in Asynchronous Automatic Assembly Systems." Discrete Event Dynamic Systems: Theory and Applications 1: 93-124.
- Kamath, M., R. Suri, and J.L. Sanders. 1988. "Analytical Performance Models for Closed-loop Flexible Assembly Systems." The International Journal of Flexible Manufacturing Systems 1: 51-84.
- Kamath, M. and N. Viswanadham. 1986. "Applications of Petri net Based Models in the Modeling and Analysis of Flexible Manufacturing Systems." In Proceedings of the 1986 IEEE International Conference on Robotics and Automation: 312-317. Washington, DC: IEEE Computer Society Press.
- Karacal, S.C. 1990. "The Development of an Integrative Structure for Discrete Event Simulation, Object Oriented Programming, and Imbedded Decision Processes." Ph.D. dissertation, Oklahoma State University.
- Kelly, F.P. 1976. "Networks of Queues." Advanced Applied Probability 8: 416-432.
- Kelly, F.P. and P.K. Pollett. 1983. "Sojourn Times in Closed Queueing Networks." Advanced Applied Probability 15: 638-656.

- Khoshnevis, B. and A. Chen. 1987. "An Automated Simulation Modeling System Based on AI Techniques." In Simulation and AI, edited by P.A. Luker and B. Birtwistle, 87-91. San Diego, CA: The Society of Computer Simulation.
- Kleijnen, J.P.C. 1979. "Regression Metamodels for Generalizing Simulation Results." IEEE Transactions on Systems, Man, and Cybernetics 9, no. 2: 93-96.
- Koenigsberg, E. 1991. "Is Queueing Theory Dead?" Omega 19, no. 2/3: 69-78.
- Kraemer, W. and M. Langenbach-Belz. 1976. "Approximate Formula for the Delay in the Queueing System GI/G/1." In Proceedings of the Eighth International Teletraffic Congress, 235-1/8.
- Kreutzer W. 1986. System Simulation Programming Styles and Languages. Reading, MA: Addison-Wesley.
- Kuehn, P.J. 1979. "Approximate Analysis of General Queueing Networks by Decomposition." IEEE Transactions Comm. 27: 113-126.
- Labetoulle, J. and G. Pujolle. 1980. "Isolation Method in a Network of Queues." Transactions on Software Engineering 6, no. 4: 373-381. IEEE.
- Law, A.M. 1986. "An Introduction to Simulation: A Powerful Tool for Analyzing Complex Manufacturing Systems." Industrial Engineering 18, no. 5: 46-63.
- Law, A.M. and S.W. Haider. 1989. "Selecting Simulation Software for Manufacturing Applications: Practical Guidelines and Software Survey." Industrial Engineering, 31, no. 5: 33-46.
- Law, A.M. and W.D. Kelton. 1991. Simulation Modeling And Analysis. 2nd ed. New York, NY: McGraw-Hill, Inc.
- Ledbetter, W.N. and J.F. Cox. 1977. "Are OR Techniques Being Used?" Industrial Engineering 9, no. 2: 19-21.
- Likic, A. and V. Zivkovic. 1989. "A Software Package for Representation and Analysis of Flexible Manufacturing Systems Based on Petri Nets." In Proceedings of the 3rd European Simulation Congress: 502-507.
- Massey, F.J., Jr. 1951. "The Kolmogorov-Smirnov Test For Goodness Of Fit." American Statistical Association Journal 46, no. 253: 68-78.
- Meisel, W.S. and D.C. Collins. 1973. "Repro-Modeling: An Approach to Efficient Model Utilization and Interpretation." IEEE Transactions on Systems, Man, and Cybernetics 3, no. 4: 349-358.

- Mitrani, I. 1982. Simulation Techniques for Discrete Event Systems. Cambridge, Great Britain: Cambridge University Press.
- Mize, J.H. 1973. "Multiple Sequence Random Number Generators." In Proceedings of the 1973 Winter Simulation Conference, edited by A.C. Hoggatt: 67-76.
- Mize, J.H. and J.G. Cox. 1968. Essentials of Simulation. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Mize, J.H., M. Kamath, and L. Leemis. 1990. "Design and Implementation of an Object-Oriented Modeling and Simulation Environment for Manufacturing Systems." A research proposal submitted to the National Science Foundation.
- Molloy, M.K. 1982. "Performance Analysis using Stochastic Petri Nets." Transactions on Computers 31, no. 9: 913-917. IEEE.
- Murata, T. 1989. "Petri Nets: Properties, Analysis and Applications." In IEEE Proceedings 77, no. 4: 541-580. IEEE.
- Nance, R.E. 1984. "Model Development Revisited." In Proceedings of the 1984 Winter Simulation Conference, edited by S. Sheppard, U.W. Pooch, and C.D. Pegden, 75-80. Piscataway, N.J.: IEEE.
- Narahari, Y. and N. Viswanadham. 1984. "Analysis and Synthesis of Flexible Manufacturing Systems using Petri Nets." In Proceedings of the First ORSA/TIMS Conference on Flexible Manufacturing Systems: 346-358. North Holland.
- Nilsson, N.J. 1980. Principles of Artificial Intelligence. Palo Alto, CA: Tioga Publishing Co.
- Nymon, J.G. 1987. "Using Analytical and Simulation Modeling For Early Factory Prototyping." In Proceedings of the 1987 Winter Simulation Conference, edited by A. Thesen, H. Grant, W.D. Kelton: 721-724.
- O'Reilly, P.J.P. and J.L Hammond. 1984. "An Efficient Simulation Technique for Performance Studies of CSMA/CD Local Networks." In IEEE Journal on Selected Areas in Communication 2, no. 1: 238-249.
- Oren, T. and K. Aytac. 1985. "Architecture of MAGEST: A Knowledge-based Modeling and Simulation System." In Simulation in Research and Development: 99-109. North-Holland, Amsterdam, The Netherlands: Elsevier Science Publishers.
- Park, S.K. and K.W. Miller. 1988. "Random Number Generators: Good Ones Are Hard To Find." Communications Of The ACM 31, no. 10: 1192-1201.

- Paul, R.J. 1991. "Recent Developments in Simulation Modelling." Journal of the Operational Research Society 42, no. 3: 217-226.
- Pegden, C.D. 1986. Introduction to SIMAN. State College, PA: Systems Modeling Corporation.
- Peterson, J.L. 1977. "Petri Nets." ACM Computing Surveys 9, no. 3: 223-252.
- Peterson, J.L. 1981. Petri Net Theory and the Modeling of Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Pritsker, A.A.B. 1986. Introduction to Simulation and SLAM II. 3rd ed. New York, NY: Halsted Press.
- Reiser, M. and S.S. Lavenberg. 1980. "Mean-Value Analysis of Closed Multichain Queueing Networks." Association for Computing Machinery Journal 27: 313-322.
- Ross, S.M. 1989. Introduction To Probability Models. 4th ed. Boston, MA: Academic Press.
- Schriber, T.J. 1987. "The Nature and Role of Simulation in the Design of Manufacturing Systems." In Simulation In CIM and Artificial Intelligence Techniques, edited by J. Reti and K.E. Wichmann: 5-18. Belgium: The Society for Computer Simulation.
- Schwetman, H.D. 1977. "Hybrid Simulation Models: A Speed-Up Technique Combining Analytical and Discrete-Event Modeling." Models of Computer Systems, edited by P.P. Spies. New York: Springer-Verlag.
- Schwetman, H.D. 1978. "Hybrid Simulation Models of Computer Systems." Communications of the ACM 21, no. 9: 718-723.
- Segal, M. and W. Whitt. 1989. "A Queueing Network Analyzer for Manufacturing." In Proceedings of the Eighth International Teletraffic Congress: 1146-1152. North Holland.
- Sevinc, S. 1988. "Automatic Simplification of Models in a Hierarchical, Modular Discrete Event Simulation Environment." Ph.D. dissertation, The University of Arizona.
- Shannon, R.E., S.S. Long, and B.P. Buckles. 1980. "Operation Research Methodologies in Industrial Engineering: A Survey." AIIE Transactions 12, no. 4.
- Shantikumar, J.G. and R.G. Sargent. 1980. "On the Approximations to the Single Server Queue." International Journal of Production Research 18: 761-773.

- Shantikumar, J.G. and R.G. Sargent. 1983. "A Unifying View of Hybrid Simulation/Analytic Models and Modeling." Operations Research 31, no. 6: 1030-1052.
- Simon, H.A., and A. Ando. 1961. "Aggregation of Variables in Dynamic Systems." Econometrica 29: 111-138.
- Solberg, J.J. 1977. A Mathematical Model of Computerized Manufacturing Systems." In Proceedings of the 4th International Conference on Production Research.
- Steel, R.G.D, and J.H. Torrie. 1980. Principles and Procedures Of Statistics. 2nd ed. New York, NY: McGraw-Hill, Inc.
- Stephens, M.A. 1974. EDF Statistics for Goodness of Fit and Some Comparisons. Journal of the American Statistical Association 69: 730-737.
- Suri, R. 1983. "Robustness of Queueing Network Formulas." Association for Computing Machinery Journal 30, no. 3: 564-594.
- Suri, R. and R.R. Hildebrant. 1984. "Modeling Flexible Manufacturing Systems Using Mean-Value Analysis." Journal of Manufacturing Systems 3, no. 1: 27-38.
- Suri, R., G.W. Diehl, and R. Dean. 1986. "Quick and Easy Manufacturing Systems Analysis Using MANUPLAN." In Proceeding of the Spring IIE Conference: 195-205. Norcross, GA: IIE.
- Suri, R. and G.W. Diehl. 1987. "Rough-cut Modeling: An Alternative to Simulation." CIM Review 3: 25-32.
- Suri, R. 1988a. "A New Perspective on Manufacturing Systems Analysis." Design and Analysis of Integrated Manufacturing Systems, edited by W.D. Compton: 118-133. Washington, DC: National Academy Press.
- Suri, R. 1988b. "RMT puts Manufacturing at the Helm." Manufacturing Engineering 100, no. 2: 41-44.
- Suri, R., J.L. Sanders, and M. Kamath. 1992. "Performance Evaluation of Production Networks." In Handbooks in Operations Research and Management Science, Vol.4: Logistics of Production and Inventory, edited by S.C. Graves, A. Rinnooy Kan, and P. Zipkin. Elsevier (forthcoming).
- Thomasian, A. and K. Gargeya. 1985. "Speeding Up Computer System Simulations Using Hierarchical Modeling." In Proceedings CMG XV International Conference '84: 845-850.

- Tijms, H.C. 1988. Stochastic Modelling and Analysis: A Computational Approach. New York, NY: John Wiley and Sons.
- Tolopka, S. and H. Schwetman. 1979. "Mix-Dependent Job Scheduling - An Application of Hybrid Simulation." In Proceedings of the NCC 48: 45-49.
- Vemuri, V. 1978. Modeling of Complex Systems. New York, NY: Academic Press.
- Walrand, J. and P. Varaiya. 1980. "Sojourn Times and the Overtaking Condition in Jacksonian Networks." Advanced Applied Probability 12: 1000-1018.
- Whitt, W. 1983. "The Queueing Network Analyzer." Bell Systems Technical Journal 62: 2779-2815.
- Whitt, W. 1984. "Open and Closed Networks of Queues." AT&T Bell Laboratory Technical Journal 63: 1911-1979.
- Wolff, R.W. 1989. Stochastic Modeling and the Theory of Queues. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Zeigler, B.P. 1984. Multifaceted Modelling and Discrete Event Simulation. Orlando, Florida: Academic Press, Inc.
- Zeigler, B.P. 1990. Object-Oriented Simulation with Hierarchical, Modular Models. San Diego, CA: Academic Press, Inc.

APPENDIXES

APPENDIX A
SMALLTALK-80 CODE FOR ENVIRONMENT
MODIFICATIONS

Introduction

This appendix contains listings of Smalltalk 80 code beginning on the next page. The listings are relevant portions of new and/or modified classes and methods that were used for hybrid metamodeling.

The listings are separated by class. Within each class, there is a header section followed by listings of methods. The header section contains the class hierarchy specification as well as the names of all instance and class variables. A comment segment concludes the header section.

The methods are divided into groups of related methods. This grouping is arbitrary but usually provides some insight as to the general intent of the methods in the group. The group headers are designated by the character string "*!classname methodsFor: groupname*". The last grouping under a method (if listed) is the group for class methods. These methods are used by the class rather than instances of the class. A good example of their use is the creation of a new instance.

Methods listings always start with the method name including any incoming parameters. The names are free form except that a colon is used to separate the parameter(s) from the name. The code itself follows Smalltalk 80 convention. Any text within the method enclosed by quotation marks is a comment. All methods terminate with an exclamation point (!).

SmallTalk-80 Code For Class: CimSimulation
--

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:37:48 pm'!

```
Simulation subclass: #CimSimulation
  instanceVariableNames: 'workFlowGenerator outputStream '
  classVariableNames: "
  poolDictionaries: "
  category: 'Cim Sim'!
```

CimSimulation comment:

'The actual Simulation model is represented by this object. '!

!CimSimulation methodsFor: 'simulation control'!

```
postponeEventForResource2: aResource by: aTime
  "SimScript cr; nextPutAll: eventQueue printString."
  | event newTime |
  event := eventQueue detect: [:ev | (ev isKindOf: WorkFlowItem)
    ifTrue: [ev resourceNeeded = aResource]
    ifFalse: [false]]
    ifNone: [nil].

  event isNil
    ifTrue: [^nil]
    ifFalse:
      [newTime := event resumptionTime + aTime.
      "SimScript cr; nextPutAll: aResource printString , ' was working with ' , event
      printString ,
      ' prev finish time: ' , event resumptionTime printString , ' new finish time: ' ,
      newTime printString."
      event resumptionTime: newTime.
      eventQueue reSort.
      newTime < Simulation active time ifTrue: [self halt]]!

postponeEventForResource: aResource by: aTime
  "SimScript cr; nextPutAll: eventQueue printString."
  | event newTime |
  event := eventQueue detect: [:ev | (ev isKindOf: WorkFlowItem)
    ifTrue: [ev resourceNeeded = aResource]
    ifFalse: [false]]
    ifNone: [nil].

  event isNil
    ifTrue:
      ["self halt."
      "SimScript cr; nextPutAll: aResource printString , ' was Waiting for Operator with ' ,
      event printString , ' Failure IGNORED'."
      ^self]
    ifFalse:
      [
      "SimScript cr; nextPutAll: 'Operator Status is ' , event hasOperator printString."
      event hasOperator = 2
      ifTrue:
        [newTime := event resumptionTime + aTime.
        "SimScript cr; nextPutAll: aResource printString , ' was in Setup
        with ' , event printString ,
        ' prev finish time: ' , event resumptionTime printString , '
        new finish time: ' , newTime printString."

```


SmallTalk-80 Code For Class: CimSimulation (continued)
--

```

event resumptionTime: newTime.
eventQueue reSort.
^event class].
(event hasOperator = 3 )
  ifTrue:
    [newTime := event resumptionTime + 100.
     "SimScript cr; nextPutAll: aResource printString , ' was Processing
with ' , event printString ,
     ' prev finish time: ' , event resumptionTime printString , '
new finish time: ' , newTime printString."
event resumptionTime: newTime.
eventQueue reSort.
^nil]]!

```

SmallTalk-80 Code For Class: ContinuousEmpiricalGrouped

'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:35:15 pm!'

```

ContinuousProbability subclass: #ContinuousEmpiricalGrouped
instanceVariableNames: 'theDataArray theCellArray aRandom dataFile '
classVariableNames: "
poolDictionaries: "
category: 'Statistics'!

```

!ContinuousEmpiricalGrouped methodsFor: 'accessing'!

```

random
  ^aRandom!

```

```

showCells
  ^theCellArray!

```

```

showData
  ^theDataArray! !

```

!ContinuousEmpiricalGrouped methodsFor: 'random sampling'!

```

next
  | u index s t1 t2 t3 |

  u := aRandom next.

  "Transcript cr; show: 'Random Number: ' , u printString , ' '."
  index := 0.
  [(theDataArray at: index + 1)
   <= u and: [(theDataArray at: index + 2)
    > u]]
    whileFalse: [index := index + 1].
  "Transcript show: 'Index: ' , index printString , ' '."
  t1 := u - (theDataArray at: index + 1).
  t2 := (theCellArray at: index + 3)
        - (theCellArray at: index + 2).
  t3 := (theDataArray at: index + 2)

```

SmallTalk-80 Code For Class: ContinuousEmpiricalGrouped (continued)

```

- (theDataArray at: index + 1).
s := (theCellArray at: index + 2) asFloat + (t1 * t2 / t3).
"Transcript show: 'Sample: ', s printString."
^s! !

!ContinuousEmpiricalGrouped methodsFor: 'private'!

getParameters
| file |
dataFile := DialogView request: 'Sampling Distribution FileName (or GLOBAL) ?' initialAnswer:
'GLOBAL'.
dataFile = 'GLOBAL'
    ifTrue:
        [theCellArray := MetaCells.
         Utils new setMetaDistribution.
         theDataArray := MetaDistribution]
    ifFalse:
        [file := ReadFile named: dataFile.
         theCellArray := SortedCollection new.
         theCellArray add: 0.0.
         theDataArray := SortedCollection new.
         140
         timesRepeat:
             [theCellArray add: file getNextNumber.
              theDataArray add: file getNextNumber]]!

getParameters: file
dataFile := ReadFile named: file.
theCellArray := SortedCollection new.
theCellArray add: 0.0.
theDataArray := SortedCollection new.
140
    timesRepeat:
        [theCellArray add: dataFile getNextNumber.
         theDataArray add: dataFile getNextNumber].
dataFile close.!

initialize
aRandom := Random new.

setDataFile: name
dataFile := name.!

usingData: x usingCells: y
theDataArray := SortedCollection new.
theDataArray addAll: x.
theCellArray := SortedCollection new.
theCellArray addAll: y.!

!ContinuousEmpiricalGrouped methodsFor: 'file manipulations'!

fileOutOn: aStream
"Put the receiver's contents on the stream"

aStream nextPutToken: 'ContinuousEmpiricalGrouped'; nextPutString: dataFile; cr! !

```

SmallTalk-80 Code For Class: ContinuousEmpiricalGrouped (continued)

```
"-----"!

ContinuousEmpiricalGrouped class
  instanceVariableNames: ""

!ContinuousEmpiricalGrouped class methodsFor: 'instance creation!'

fileInFrom: aStream
  "Construct an instance from a Stream"

  | temp |
  temp := self new.
  temp initialize.
  temp getParameters: aStream getNextString.
  ^temp!

usingData: aCollection usingCells: bCollection
  "Input is a Collection"

  | t |
  t := self new.
  t initialize.
  t usingData: aCollection usingCells: bCollection.
  ^t!
```

SmallTalk-80 Code For Class: Exponential
--

```
'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:34:29 pm!'

ContinuousProbability subclass: #Exponential
  instanceVariableNames: 'mu aRandom '
  classVariableNames: ""
  poolDictionaries: ""
  category: 'Statistics'!

!Exponential methodsFor: 'accessing'!

mean
  ^1.0/mu!

next
  "This is a general random number generation method for any probability law; use the (0,1) uniformly
  distributed random variable U as the value of the law's distribution function. Obtain the next random value and then
  solve for the inverse. The inverse solution is defined by the subclass."

  ^self inverseDistribution: aRandom next!

random
  ^aRandom!

variance
  ^1.0/(mu*mu)!!
```

SmallTalk-80 Code For Class: Exponential (continued)
--

!Exponential methodsFor: 'probability functions'!

density: x

 x > 0.0

 ifTrue: [^mu * (mu*x) negated exp]

 ifFalse: [^0.0]!

distribution: anInterval

 anInterval last <= 0.0

 ifTrue: [^0.0]

 ifFalse: [^1.0 - (mu * anInterval last) negated exp - (anInterval first > 0.0 ifTrue: [self

distribution: (0.0 to: anInterval first)] ifFalse: [0.0)]! !

!Exponential methodsFor: 'file manipulations'!

fileOutOn: aStream

 "Put the receiver's contents on the stream"

 aStream nextPutToken: 'Exponential'; nextPutNumber: mu; cr! !

!Exponential methodsFor: 'private'!

getParameters

 | muString |

 mu isNil

 ifTrue: [muString := ""]

 ifFalse: [muString := (1.0 / mu) printString].

 mu := (DialogView request: 'Mean of exponential?' initialAnswer: muString) asNumber.

 [mu > 0]

 whileFalse: [mu := (DialogView request: 'Mean of exponential should exceed 0!!!!'

initialAnswer: mu printString) asNumber].

 mu := 1.0 / mu!

initialize

 aRandom := Random new.!

inverseDistribution: x

| y |

 y := x.

 [y = 0.0] whileTrue: [y := aRandom next].

 ^ y ln negated / mu!

setParameter: p

 mu _ p! !

 "-----" !

Exponential class

 instanceVariableNames: 'aRandom' !

!Exponential class methodsFor: 'instance creation'!

fileInFrom: aStream

SmallTalk-80 Code For Class: Exponential (continued)
--

"Construct an instance from a Stream"

```
| temp |
temp := self new setParameter: aStream getNextNumber.
^temp initialize!
```

mean: p

```
laSeed!
aSeed := DialogView request: 'Seed?'.
aRandom := Random fromGenerator: 8 seededWith: aSeed.
^self parameter: 1.0/p!
```

mean: p deviation: aDummyNumber

```
^self parameter: 1.0/p!
```

parameter: p

```
p > 0.0
ifTrue: [^self new setParameter: p]
ifFalse: [self error: "The probability parameter must be greater than 0.0"!]
```

SmallTalk-80 Code For Class: Gamma

'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:34:16 pm'

Exponential subclass: #Gamma

```
instanceVariableNames: 'alpha '
classVariableNames: "
poolDictionaries: "
category: 'Statistics'!
```

!Gamma methodsFor: 'random sampling'!

next

"This routine generates Gamma variates using the method of Ahrens and Dieter (1974) when alpha < 1 and Cheng' (1977) when alpha > 1"

```
| r s b p y a c v z w flag value |
alpha < 1.0
  ifTrue:
    [b := alpha + 1 exp / 1 exp.
     flag := 0.
     [flag == 0]
      whileTrue:
        [r := aRandom next.
         s := aRandom next.
         p := b * r.
         p <= 1.0
          ifTrue:
            [y := p raisedTo: 1.0 / alpha.
             s <= y negated exp ifTrue: [flag := 1]]
          ifFalse:
            [y := (b - p / alpha) ln negated.
```

SmallTalk-80 Code For Class: Gamma (continued)
--

```

s <= (y raisedTo: alpha - 1) ifTrue: [flag := 1]].
value := y / mu]
ifFalse:
  [a := 1.0 / (2.0 * alpha - 1.0) sqrt.
  b := alpha - 4 ln.
  c := alpha + (1.0 / a).
  z := 1.0.
  w := 0.
  [w + 2.50408 - (4.5 * z) <= 0.0 & (w <= z ln)]
  whileTrue:
    [r := aRandom next.
    s := aRandom next.
    v := (r / (1.0 - r)) ln * a.
    y := v exp * alpha.
    z := r squared * s.
    w := c * v + b - y].
value := y / mu].
^value! !

!Gamma methodsFor: 'accessing'!

mean
  ^alpha / mu!

variance
  ^alpha / (mu*mu)! !

!Gamma methodsFor: 'probability functions'!

density: x
  | t |
  [x > 0.0]
  ifTrue:
    [t _ mu * x.
    ^(mu raisedTo: alpha)
    / (self gamma: alpha) * (x raisedTo: alpha - 1.0) * t negated exp]
  ifFalse: [^0.0)! !

!Gamma methodsFor: 'private'!

getParameters
  | numbers messageList initialList muString alphaString |
  mu isNil
    ifTrue: [muString := "]
    ifFalse: [muString := (1.0/mu) printString].
  alpha isNil
    ifTrue: [alphaString := "]
    ifFalse: [alphaString := alpha printString].
  messageList := Array with: 'Gamma processes' with: 'mean'.
  initialList := Array with: alphaString with: muString.
  numbers := DialogView requestList: messageList initialValues: initialList.
  alpha := (numbers at: 1) asNumber.
  mu := 1/((numbers at: 2) asNumber).!

inverseDistribution: x

```

SmallTalk-80 Code For Class: Gamma (continued)
--

```

self error: 'Gamma does not implement inverseDistribution!'

shape: events scale: mmean
  alpha := events.
  self setParameter: 1.0 / mmean! !

!Gamma methodsFor: 'file manipulations!'

fileOutOn: aStream
  "Put the receiver's contents on the stream"

  aStream nextPutToken: 'Gamma'; nextPutNumber: mu; nextPutNumber: alpha; cr! !
  "-----"!

Gamma class
  instanceVariableNames: ""

!Gamma class methodsFor: 'instance creation!'

fileInFrom: aStream
  "Construct an instance from a Stream"

  | temp events mean |
  mean := aStream getNextNumber.
  events := aStream getNextNumber.
  temp := self new shape: events scale: (1/mean).
  ^temp initialize!

shape: q scale: p
  "Gamma with q processes and a mean of p"

  q > 0.0
    ifTrue: [p > 0.0
      ifTrue: [^self new shape: q scale: p]
      ifFalse: [self error: 'the rate for the Gamma must be greater than 0.0']]
    ifFalse: [self error: 'the number of events for the Gamma must be greater than 0.0']! !

```

SmallTalk-80 Code For Class: HypoExponential2

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:34:44 pm'!

```

Exponential subclass: #HypoExponential2
  instanceVariableNames: 'mu1 mu2 '
  classVariableNames: ""
  poolDictionaries: ""
  category: 'Statistics!'

!HyperExponential2 methodsFor: 'random sampling!'

next
  "This routine generates HyperExponential (2 stage) variates by summing
  two exponential variates "

```

SmallTalk-80 Code For Class: HypoExponential2 (continued)

```

| r s      y1 y2 |
r := aRandom next.
[r = 0] whileTrue: [r := aRandom next].
s := aRandom next.
[s = 0] whileTrue: [s := aRandom next].
y1 := r ln negated / mu1.
y2 := s ln negated / mu2.
^y1 + y2! !

!HyperExponential2 methodsFor: 'accessing'!

mean
    ^1 / mu1 + (1 / mu2)!

variance
    ^1 / (mu1 * mu1) + (1 / (mu2 * mu2))! !

!HyperExponential2 methodsFor: 'probability functions'!

density: x
    | t1 t2 |
    [x > 0.0]
        ifTrue:
            [t1 := mu1 * x.
             t2 := mu2 * x.
             ^mu1 * mu2 / (mu2 - mu1) * (t1 negated exp - t2 negated exp)]
        ifFalse: [^0.0]! !

!HyperExponential2 methodsFor: 'private'!

getParameters
    | numbers messageList initialList  mu1String mu2String |
    mu1 isNil
        ifTrue: [mu1String := ""]
        ifFalse: [mu1String := (1.0 / mu1) printString].
    mu2 isNil
        ifTrue: [mu2String := ""]
        ifFalse: [mu2String := (1.0 / mu2) printString].
    messageList := Array with: 'HyperExponential (2 Stage) mean1' with: 'mean2'.
    initialList := Array with: mu1String with: mu2String.
    numbers := DialogView requestList: messageList initialValues: initialList.
    mu1 := 1 / (numbers at: 1) asNumber.
    mu2 := 1 / (numbers at: 2) asNumber!

inverseDistribution: x
    self error: 'HyperExponential (2 Stage) does not implement inverseDistribution'!

mu1: num1 mu2: num2
    "set HyperExponential (2 Stage) parameters"

    mu1 := 1.0 / num1.
    mu2 := 1.0 / num2!

shape: events scale: mmean

```


SmallTalk-80 Code For Class: HypoExponential2 (continued)

```

alpha := events.
self setParameter: 1.0 / mmean! !

!HypoExponential2 methodsFor: 'file manipulations'

fileOutOn: aStream
    "Put the receiver's contents on the stream"

    aStream nextPutToken: 'HypoExponential2'; nextPutNumber: mu1; nextPutNumber: mu2; cr! !
    "-----"!

HypoExponential2 class
    instanceVariableNames: ""

!HypoExponential2 class methodsFor: 'instance creation'

fileInFrom: aStream
    "Construct an instance from a Stream"

    | temp num1 num2 |
    num1 := aStream getNextNumber.
    num2 := aStream getNextNumber.
    temp := self new mu1: 1 / num1 mu2: 1 / num2.
    ^temp initialize!

shape: q scale: p
    "Gamma with q processes and a mean of p"

    q > 0.0
        ifTrue: [p > 0.0
            ifTrue: [^self new shape: q scale: p]
            ifFalse: [self error: 'the rate for the Gamma must be greater than 0.0']]
        ifFalse: [self error: 'the number of events for the Gamma must be greater than 0.0']! !

```

SmallTalk-80 Code For Class: Operator

'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:36:19 pm'!

```

WorkStation subclass: #Operator
    instanceVariableNames: 'mhLocations timeMatrix '
    classVariableNames: ""
    poolDictionaries: ""
    category: 'Cim Resources'

!Operator methodsFor: 'initialize-release'

initializeWithName: aString andAmount: aNumber
    name := aString.
    wsAmountAvailable := aNumber.

    "mhLocations := OrderedCollection new.
    aNumber timesRepeat: [mhLocations add: aLocation]."
```

SmallTalk-80 Code For Class: Operator (continued)

```

wsQueueController := QueueController new.
wsProcessingTimes := ObsTrackedNumber new.
wsUtilization := TimeTrackedNumber new.

"WorkflowItem withMaterialHandling.
timeMatrix := aTimeMatrix"!

initializeWithName: aString andAmount: aNumber location: aLocation timeInfo: aTimeMatrix
self error: 'This method is obsolete for Operator'.
name := aString.
wsAmountAvailable := aNumber.
mhLocations := OrderedCollection new.
aNumber timesRepeat: [mhLocations add: aLocation].
wsQueueController := QueueController new.
wsProcessingTimes := ObsTrackedNumber new.
wsUtilization := TimeTrackedNumber new.
WorkflowItem withMaterialHandling.
timeMatrix := aTimeMatrix! !

!Operator methodsFor: 'accessing'!

mhLocations
self error: 'This method is obsolete for Operator'.
^mhLocations!

timeFrom: aLocation to: another
"Pass the message to the time matrix"

self error: 'This method is obsolete for Operator'.
^timeMatrix timeFrom: aLocation to: another! !

!Operator methodsFor: 'task language'!

provideServices
"provide operator services to the next job in queue"
|waiting wfi|
[wsQueueController inputQueueEmpty not and: [ wfi := wsQueueController next.
1 <= wsAmountAvailable]]
whileTrue:
[waiting := wsQueueController inputQueueRemove: wfi .
wsAmountAvailable := wsAmountAvailable - 1.
waiting resume.!]

provideServiceTo: aWFI
"This wfi needs to be serviced. Put into the queue, and provide
a server if possible"

wsQueueController addToInputQueue: aWFI.
"SimScript cr; nextPutAll: aWFI printString , ' needs workcenter ' , aWFI location name , '' ,
name , ' at ' , Simulation active time printString."
self provideServices.
aWFI pause.
"SimScript cr; nextPutAll: aWFI printString , ' acquired workcenter ' , aWFI location name , '' ,
name , ' at ' , Simulation active time printString."

```

SmallTalk-80 Code For Class: Operator (continued)

```

wsUtilization equals: wsUtilization value + 1.!

release: anAmount
    "release anAmount of the Operator"

    wsUtilization equals: wsUtilization value - anAmount.
    self produce: anAmount!

release: anAmount at: aLocation
    "release anAmount of the material handler at aLocation"

    self error: 'This method is obsolete for Operator'.
    wsUtilization equals: wsUtilization value - anAmount.    "SimScript cr; nextPutAll: name, ' is at ',
aLocation name."
    mhLocations addLast: aLocation.
    self produce: anAmount!

releaseBy: aWFI
    "release anAmount of the Operator"

    wsUtilization equals: wsUtilization value - 1.
    self produce: 1.
    "SimScript cr; nextPutAll: aWFI printString , ' released workcenter ',(aWFI location name),' ,
    name , ' at ', Simulation active time printString."!!

!Operator methodsFor: 'decisions!'

chooseOperator
    "Pick one of the material handlers from the
    waiting material handler at different locations"

    self error: 'This method is obsolete for Operator'.
    ^mhLocations removeFirst! !
    "-----"!

Operator class
    instanceVariableNames: "!

!Operator class methodsFor: 'instance creation!'

newWithName: aString andAmount: aNumber

"Create a new OPERATOR for this WorkCenter"

    ^self new initializeWithName: aString andAmount: aNumber!

newWithName: aString andAmount: aNumber location: aLocation timeInfo: aTimeMatrix

"Create a new transport device at this location"
self error: 'This method is obsolete'.
    ^self new initializeWithName: aString andAmount: aNumber location: aLocation timeInfo: aTimeMatrix!
!

```

SmallTalk-80 Code For Class: Plant

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:35:33 pm'

Object subclass: #Plant

```
instanceVariableNames: 'workCenters transportDevice buffers machineLocDictionary routingDictionary
bom disposer mfgController '
classVariableNames: 'ActivePlant '
poolDictionaries: "
category: 'Cim Controllers'!
```

!Plant methodsFor: 'decisions'!

whatShouldIDo: aWFI

"This wfi needs a decision on what its next operation is"

"THIS IS A MODIFIED VERSION OF THE METHOD TO IMPLEMENT
PREFERENTIAL QUEUE SELECTION"

| queueLength operation machine workCenter q2 op2 q3 op3 |

queueLength := 987654.

aWFI areYouDone

ifTrue: [self error: 'This wfi is done.']

ifFalse: [(aWFI routing at: aWFI currentStage)

do:

[:op |

"This is the list of current alternate operations"

machine := op machine.

workCenter := machineLocDictionary at: machine.

"Transcript cr; show: op machine , ' '; show: (self resourceName:

op machine) queueLength printString."

op machine = 'pack1'

ifTrue:

[q2 := queueLength := (workCenter

op2 := op].

op machine = 'pack2'

ifTrue:

[q3 := queueLength := (workCenter

op3 := op].

(workCenter resourceName: op machine) queueLength <

queueLength

ifTrue:

[queueLength := (workCenter resourceName:

op machine) queueLength.

operation := op]]].

"Logic For PREFERRED QUEUEING STRATEGY."

"q3 isNil ifFalse: [q3 - 4 < q2

ifTrue: [operation := op3]

ifFalse: [operation := op2]]."

"Logic For SHORTEST QUEUE STRATEGY."

q3 isNil ifFalse: [q3 >= q2

ifTrue: [operation := op2]

ifFalse: [operation := op3]].

"Transcript cr; show: 'Routed to: ', operation machine."

^operation!

SmallTalk-80 Code For Class: Plant (continued)
--

whatShouldIDoV1: aWFI

"This wfi needs a decision on what its next operation is"
 "THIS IS A COPY OF THE ORIGINAL VERSION OF THE METHOD"

lqueueLength operation machine workCenter !

```

queueLength := 987654.
aWFI areYouDone ifTrue: [self error: 'This wfi is done. ]
ifFalse: [(aWFI routing at: aWFI currentStage) "This is the list of
current alternate operations"
do: [:op|
machine := op machine.
workCenter := machineLocDictionary at: machine .

(((workCenter resourceNamed: op machine) queueLength) < queueLength)
ifTrue: [queueLength := (workCenter resourceNamed: op machine) queueLength.
operation := op]
]].
^operation!
  
```

SmallTalk-80 Code For Class: Random

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:33:57 pm'

Stream subclass: #Random

```

instanceVariableNames: 'seed increment modulus fmodulus multiplier '
classVariableNames: 'DefaultGenerator Increments MaxGenerator Moduli Multipliers '
poolDictionaries: "
category: 'Magnitude-Numbers'!
  
```

Random comment:

'An instance of class Random provides an endless supply of random numbers.

We produce a uniform deviate in the half-open interval [0.0,1.0) using a linear congruential generator.

seed increment modulus fmodulus multiplier

See "Numerical Recipes" (W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling; Cambridge University Press 1986), pp. 191-199.

Instance Variables:

```

seed          <Integer> the first of the series.
increment     <Integer>
modulus       <Integer>
fmodulus     <Float>
multiplier    <Integer>
              The recurrence parameters.
  
```

Class Variables:

```

DefaultGenerator <Integer> used to choose a generator if the client
                  doesn't select one.
Increments       <Array of: Integer>
MaxGenerator     <Integer> Highest numbered generator permitted;
  
```

SmallTalk-80 Code For Class: Random (continued)

```

                this is the length of the 3 arrays named below.
Moduli          <Array of: Integer>
Multipliers     <Array of: Integer>
                Constants for the recurrence parameters. generator:
                selects from these, e.g. generator #2 uses:
                {Increments at: 2, Moduli at:2, Multipliers at:
2).!

!Random methodsFor: 'accessing'!

contents
    "Random numbers do not have a contents so provide
    an error notification."

    ^self shouldNotImplement!

flush
    "Random numbers do not need to flush."

    ^self shouldNotImplement!

next
    "Answer the next random number."

    ^self step asFloat / fmodulus!

nextPut: anObject
    "Random numbers do not implement nextPut: so provide an
    error notification."

    ^self shouldNotImplement! !

!Random methodsFor: 'testing'!

atEnd
    "Answer false that the stream is not at an end."

    ^false! !

!Random methodsFor: 'private'!

generator: aSmallInteger
    "Chooses a parameter triplet."

    | generatorIndex |
    generatorIndex := aSmallInteger.
    generatorIndex < 1 | (generatorIndex > MaxGenerator)
        ifTrue:
            [self notify: 'No such generator; proceed for generator #1'.
            generatorIndex := 1].
    increment := Increments at: generatorIndex.
    modulus := Moduli at: generatorIndex.
    fmodulus := modulus asFloat.
    multiplier := Multipliers at: generatorIndex!

```

SmallTalk-80 Code For Class: Random (continued)

```

seed
    "Return the current seed value"

    ^seed!

seed: aSmallInteger
    "Initialize the first random number."

    seed := aSmallInteger \modulus!

setSeed
    "Initialize the first random number."

    seed := Time millisecondClockValue bitAnd: 65535
        "Time millisecondClockValue gives a large integer; I only want the lower 16 bits.!"

step
    "Produce the next random seed."

    "Transcript show: ' ',(multiplier printString),'-',(seed printString),' '!"
    seed := seed * multiplier + increment \modulus.
    "Transcript show: (seed printString),' '!"
    ^seed! !

"-----"!

Random class
    instanceVariableNames: "!"

!Random class methodsFor: 'class initialization'!

initialize
    "Set the recurrence parameter constants."
    "These values are appropriate for
    SmallInteger maxVal = ((2 raisedToInteger: 29) - 1)"
    "After changing the DefaultGenerator, Execute-> Random initialize"
    "

    IMPORTANT NOTE:
    Generator 8 is Park & Millers Minimal Standard LCG, (Park & Miller, Comm. of ACM, Oct. 88). This
generator is
    one of the (if not THE) best RNGs known. It can be validated (according to Park & Miller) by
demonstrating
    that the 10,000th seed generated is 10436118065. This has been done by DBP in ST80V40 running on a
    386 PC on 8/14/91. Unless the you have an RNG proven to be better than Park & Miller, DO NOT
change the
    default generator or override it !! The generator can be seeded with ANY integer between 1 and
    2147483646.
    Park & Miller suggest a student's SSN as a good seed.
    "

    DefaultGenerator := 8.
    MaxGenerator := 8.
    Moduli := #(120050 214326 244944 233280 175000 121500 145800 2147483647 ).
    Multipliers := #(2311 1807 1597 1861 2661 4081 3661 16807 ).

```

SmallTalk-80 Code For Class: Random (continued)

```
Increments := #(25367 45289 51749 49297 36979 25673 30809 0)!!
```

```
!Random class methodsFor: 'instance creation'
```

```
fromGenerator: g seededWith: s
```

```
    "Answer a new random number generator."
```

```
    | r |
```

```
    r := self basicNew.
```

```
    r generator: g.
```

```
    r seed: s.
```

```
    ^r!
```

```
new
```

```
    "Answer a new random number generator, seeded from the time-of-day. "
```

```
    "The simple, naive interface..."
```

```
    ^self fromGenerator: DefaultGenerator
```

```
        seededWith: Time millisecondClockValue! !
```

```
Random initialize!
```

SmallTalk-80 Code For Class: SimModel

```
'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:37:20 pm'!
```

```
Model subclass: #SimModel
```

```
    instanceVariableNames: 'plants plantSelection plantMenu workCenterSelection workCenterMenu
workStationSelection workStationMenu plantButtonSelection parentObject currentObject plantDisplayCollection
distanceSelection failSelection repairSelection failRes
et repairReset bom routing viewSelection routingPartSelection operationSelection alternateSelection
processTimeReset processTimeSelection setUpTimeReset setUpTimeSelection disposalDecisions
arrivalDistributions disposalSelection arrivalTimeReset arrivalTi
meSelection wsSelection histogramSelection histograms histogramNameList histogramStationList
histogramPartList providedHistogramList '
    classVariableNames: 'TextMenu '
    poolDictionaries: "
    category: 'Cim Interface'!
```

```
addMetaWorkStation
```

```
    "Add a new meta workstation to this workCenter"
```

```
    | |
```

```
    parentObject := workCenterSelection.
```

```
    distanceSelection := nil.
```

```
    failReset := repairReset := false.
```

```
    currentObject := InterfaceWorkStation newWithName: 'meta'.
```

```
    workCenterSelection addWorkStation: currentObject.
```

```
    currentObject outputQCapacity: 0.
```

```
    currentObject inputQCapacity: 0.
```

```
    parentObject := nil.
```

```
    currentObject := nil.
```

```
    self changed: #workStationName!
```


SmallTalk-80 Code For Class: SimModel (continued)

metaReplace

"This method replaces all the selected workcenter's workstations with a meta workstation and updates the routing"

```

| workstation wsKeys routKeys operList rout firstTrue test operation dummy |
(MetaWcList includes: workCenterSelection) ifFalse: [DialogView notify:
  (('No MetaModel Has Been Defined For This WorkCenter\
That!!') withCRs).
  Sorry About
  ^self].
routKeys := self routing keys.
routKeys do:
  [:rt |
  rout := self routing at: rt.
  firstTrue := true.
  operList := rout firstOperationList.
  operList do:
    [:op |
    test := workCenterSelection includesResourceFor: op machine.
    "Transcript cr; show: op machine , ' ', test printString."
    test
      ifTrue:
        [firstTrue
          ifTrue:
            ["Transcript cr; show: 'Implement Add meta'."
            MetaService := op processTime mean.
            operation := Operation new.
            processTimeReset := setUpTimeReset := false.
            currentObject := operation.
            operation machine: 'meta'.
            rout addOperation: operation before: op.
            currentObject := nil.
            self changed: #operationName.
            self changed: #alternateName.
            firstTrue := false].
            "Transcript cr; show: 'Implement Delete operation'."
            rout removeOperation: op.
            operationSelection := nil.
            alternateSelection := nil.
            self changed: #operationName.
            self changed: #alternateName]
          ifFalse: ["Transcript cr; show: 'Do Nothing'"]]].
wsKeys := workCenterSelection resources keys.
wsKeys do:
  [:ws |
  workstation := workCenterSelection resources at: ws.
  workstationSelection := nil.
  workCenterSelection removeWorkStation: workstation.
  self changed: #workStationName].
self addMetaWorkStation.
dummy := Utils new.
dummy getMetaParameters.
"dummy setMetaDistribution."!

```

workCenterMenu

"Answer an ActionMenu of operations on workCenters that is to be displayed"

SmallTalk-80 Code For Class: SimModel (continued)

```

when the operate menu button is pressed."

plantSelection isNil ifTrue: [^workCenterMenu := nil].
workCenterSelection isNil
    ifTrue: [workCenterMenu _ ActionMenu
        labels: 'add a workCenter' withCRs
        lines: #()
        selectors: #(#addWorkCenter)]
    ifFalse: [workCenterMenu _ ActionMenu
        labels: 'Add a workCenter\modify-
review\remove\rename\Meta Replace' withCRs
        lines: #(2 4)
        selectors: #(#addWorkCenter #modifyWorkCenter

#removeWorkCenter #renameWorkCenter
    #metaReplace)].
    ^workCenterMenu!

carryOutExperimentWith: anArray
    "Build the model to be simulated from parts of the array"

| controller destinations cumProbs sim newPlant plant workCenter workStation mh timeMatrix wf index
route routingDictionary |

ProbabilityDistribution initializeWithSeed: (anArray at: 1) asNumber.
controller := MfgController new.
disposalDecisions
    keysAndValuesDo:
        [:part :disposals |
        | cumProb |
        destinations := OrderedCollection new.
        cumProbs := OrderedCollection new.
        cumProb := 0.0.
        disposals
            keysAndValuesDo:
                [:dest :percent |
                destinations add: dest.
                cumProbs add: (cumProb := cumProb + (percent / 100))].

        controller
            putDisposalDecisionFor: part
            destinations: destinations
            cumProbability: cumProbs].

sim := CimSimulation new.
newPlant := Plant new.
newPlant activate.
plant := self plant.
plant workCenters do:
    [:wc |
    workCenter := WorkCenter newWithName: wc name.
    wc resources do:
        [:res |
        (res isKindOf: InterfaceAssemblyStation)
            ifTrue: [workStation := workCenter addAssemblyStation: res name]
            ifFalse:
                [workStation := workCenter addWorkStation: res name.

```

SmallTalk-80 Code For Class: SimModel (continued)

```

res inputQCapacity > 0 ifTrue: [workStation inputQueueCapacity:
res inputQCapacity].
0].
res outputQCapacity > 0 ifTrue: [workStation
outputQueueCapacity: res outputQCapacity].
res outputQCapacity = -1 ifTrue: [workStation
outputQueueCapacity: 0]].
].
mh := wc materialHandler.
mh isNil | (mh = #None)
  ifFalse:
    [
      timeMatrix := wc getTimeMatrix.
      mh = #AGV ifTrue: [workCenter
        addMaterialHandler: 'transport' , wc name
        amount: wc mhQuantity
        timeInfo: timeMatrix].
      mh = #Conveyor ifTrue: [workCenter addConveyor: 'transport' , wc name
timeInfo: timeMatrix]].
      newPlant addWorkCenter: workCenter].
mh := plant materialHandler.
mh isNil | (mh = #None)
  ifFalse:
    [
      timeMatrix := plant getTimeMatrix.

      mh = #AGV ifTrue: [newPlant
        addMaterialHandler: 'transport' , plant name
        amount: plant mhQuantity
        timeInfo: timeMatrix].
      mh = #Conveyor ifTrue: [newPlant addConveyor: 'transport' , plant name timeInfo:
timeMatrix]].
      routingDictionary := self routing copy.
      self bom partList do: [:part | part areYouAnAssembly
        ifTrue:
          [route := routingDictionary at: part name.
            route currentAlternates do: [:op | (newPlant resourceNamed: op machine)
              assemblyTime: op processTime;
              assemblyName: part name].
            route getRidOfFirstOperation]].

      sim activate.
      sim outputStream: (ResultScript := TextStream on: (String new: 1024)).
      newPlant routingDictionary: routingDictionary.
      arrivalDistributions
        keysAndValuesDo:
          [:part :dist |
            dist initialize.
            wf := WorkflowGenerator name: part arrivalDistribution: dist.
            sim addWorkflowGenerator: wf].
      providedHistogramList := OrderedCollection new.
      histograms do:
        [:name |
          index := histogramNameList indexOf: name.
          index <= histogramStationList size

```

SmallTalk-80 Code For Class: SimModel (continued)

```

        ifTrue: [providedHistogramList add: (newPlant resourceName: (histogramStationList
at: index) name) provideHistogram]
        ifFalse: [providedHistogramList add: (WorkflowItem provideHistogramForPart:
(histogramPartList at: index - histogramStationList size) name)]];
    sim traceOnAt: (anArray at: 4) asNumber.
    newPlant mfgController: controller.
    newPlant bom: self bom.
    SimulationNumber := SimulationNumber + 1.
    "SimulationNumber inspect."
    sim outputStream nextPutAll: 'Plant: ', (plant name), '; Log#=', SimulationNumber printString, '; seed=',
(anArray at:1), '; term=', (anArray at:2), '; clear=', (anArray at:3);cr.
    plant workCenters do:
        [:wc |

            wc resources do:
                [:res |

                    res failureDistribution isNil
                    ifFalse:
                        [res failureDistribution: res failureDistribution
repairDistribution: res repairDistribution]]].

    sim startUp.
    sim clearStatisticsAt: (anArray at: 3) asNumber.
    "Cursor execute showWhile: [[sim time < (anArray at: 2) asNumber]
whileTrue: [sim proceed]]."
    Cursor execute showWhile: [[ ((providedHistogramList at:1) list) size < (anArray at: 2) asNumber]
whileTrue:
        [sim proceed: (anArray at: 1)]].
    sim finishUp.
    Transcript endEntry.!
```

SmallTalk-80 Code For Class: Triangular

'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:35:02 pm'!

```

ContinuousProbability subclass: #Triangular
instanceVariableNames: 'lowerBound mode upperBound aRandom '
classVariableNames: "
poolDictionaries: "
category: 'Statistics'!
```

!Triangular methodsFor: 'probability distributions'!

```

density: x
    "not Finished"!!
```

!Triangular methodsFor: 'accesssing'!

```

mean
    ^lowerBound + upperBound + mode / 3.0!
```

```

next
    "This is a general random number generation method for any probability law; use the (0,1)
```

SmallTalk-80 Code For Class: Triangular (continued)

uniformly distributed random variable U as the value of the law's distribution function. Obtain the next random value and then solve for the inverse. The inverse solution is defined by the subclass."

```

^self inverseDistribution: aRandom next!

random
  ^aRandom!

variance
  ^lowerBound squared + upperBound squared + mode squared - (lowerBound * upperBound) -
  (lowerBound * mode) - (upperBound * mode) / 18.0! !

!Triangular methodsFor: 'private'!

from: a to: b mode: c
  lowerBound := a.
  upperBound := b.
  mode := c!

getParameters
  | numbers messageList initialList lbString modeString ubString |
  lowerBound isNil
    ifTrue: [lbString := ""]
    ifFalse: [lbString := lowerBound printString].
  mode isNil
    ifTrue: [modeString := ""]
    ifFalse: [modeString := mode printString].
  upperBound isNil
    ifTrue: [ubString := ""]
    ifFalse: [ubString := upperBound printString].
  messageList := Array with: 'Triangular Lower Bound' with: 'Mode' with: 'Upper Bound'.
  initialList := Array with: lbString with: modeString with: ubString.
  numbers := DialogView requestList: messageList initialValues: initialList.
  lowerBound := (numbers at: 1) asNumber.
  mode := (numbers at: 2) asNumber.
  upperBound := (numbers at: 3) asNumber.!

initialize
  aRandom := Random new!

inverseDistribution: x
  | v |
  v := mode - lowerBound / (upperBound - lowerBound).
  x <= v
    ifTrue: [^lowerBound + ((upperBound - lowerBound) * (v * x) sqrt)]
    ifFalse: [^lowerBound + ((upperBound - lowerBound) * (1.0 - (1.0 - v * (1.0 - x)) sqrt))! !

!Triangular methodsFor: 'file manipulations'!

fileOutOn: aStream
  "Put the receiver's contents on the stream"

  aStream nextPutToken: 'Triangular'; nextPutNumber: lowerBound; nextPutNumber: mode;
  nextPutNumber: upperBound; cr! !

```

SmallTalk-80 Code For Class: Triangular (continued)

```
"-----"
```

```
Triangular class
```

```
instanceVariableNames: ""
```

```
!Triangular class methodsFor: 'instance creation'!
```

```
fileInFrom: aStream
```

```
"Construct an instance from a Stream"
```

```
ltemp lb mode ub l
```

```
lb := aStream getNextNumber.
```

```
mode := aStream getNextNumber.
```

```
ub := aStream getNextNumber.
```

```
temp := self new
```

```
from: lb
```

```
to: ub
```

```
mode: mode.
```

```
^temp initialize.!
```

```
from: a to: b mode: c
```

```
b > a
```

```
ifTrue: [c > a
```

```
ifTrue: [b > c
```

```
ifTrue: [^self new
```

```
from: a
```

```
to: b
```

```
mode: c]
```

```
ifFalse: [^self error: 'Bad range on Triangular mode']]
```

```
ifFalse: [^self error: 'Bad range on Triangular mode']]
```

```
ifFalse: [^self error: 'Bad range on Triangular']! !
```

SmallTalk-80 Code For Class: Utils

```
'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 26 December 1991 at 4:23:02 pm'!
```

```
Object subclass: #Utils
```

```
instanceVariableNames: "
```

```
classVariableNames: "
```

```
poolDictionaries: "
```

```
category: 'dbp'!
```

```
!Utils methodsFor: 'misc'!
```

```
fileWriteplant: aName run: aLetter rho: aRho model: theModel
```

```
"file writing routine to save TIS values for dbp dissertation"
```

```
ltemp fileName aFileStream n fileName2 bFileStream fileName3 cFileStream l
```

```
"temp := (TrackedNumberWithCollection allInstances at:1) list."
```

```
temp := (theModel providedHistogramList at:1) list.
```

```
fileName := aName,'t.',aLetter,aRho.
```

SmallTalk-80 Code For Class: Utils (continued)
--

```

aFileStream := (Filename named: fileName) writeStream.
n:=0.
temp do: [:x| n:=n+1. aFileStream nextPutNumber: x; space. n\7=0 ifTrue:[aFileStream cr.]].
aFileStream close.
Transcript cr; show: fileName, ' TIS file written'.

```

```

fileName2 := aName,'s.',aLetter,aRho.
bFileStream := (Filename named: fileName2) writeStream.
bFileStream nextPutAll: ResultScript contents string.
bFileStream close.
Transcript cr; show: fileName2, ' STATS file written'.

```

```

Transcript cr; show: "Transcript clear"; cr.!

```

getMetaParameters

"Input the meta workstation sampling parameters, all of which are stored in global variables
MetaRhos is a sorted collection of input rho values
MetaCells is a sorted collection of the upper boundary of the distribution cells
MetaValues is an ordered collection of sorted collections containing the values
associated with each rho value."

```

| file dataFile rho temp max string |
MetaRhos := SortedCollection new.
MetaCells := SortedCollection new.
MetaValues := OrderedCollection new.
file := DialogView request: 'Meta File Name?'.
Cursor read showWhile: [
dataFile := ReadFile named: file.
max := dataFile getNextNumber .
max + 1 timesRepeat: [MetaCells add: dataFile getNextNumber].
rho := dataFile getNextNumber.
[rho = 0]
whileFalse:
[MetaRhos add: rho .
temp := SortedCollection new.
max timesRepeat: [temp add: dataFile getNextNumber].
MetaValues add: temp.
rho := dataFile getNextNumber].].
string := 'Meta Rhos Have Been Added For: '.
1 to: (MetaRhos size) do: [:i| string := string, \, (MetaRhos at:i) printString.].
DialogView notify: (string withCRs asComposedText centered).
DialogView notify: (('DON'T Forget To Specify the MetaModel SERVICE and SETUP Distributions\,
'BEFORE You Attempt to Run The Model !!!!!') withCRs

```

asComposedText centered)!

setMetaDistribution

"Calculate the Meta Distribution based on the global Meta values and the user supplied rho value"

```

| rho trueRhos bracketed max k lo hi interp loValues hiValues result |
rho := (DialogView request: 'Rho Value To Initialize MetaModel?') asNumber.
trueRhos := OrderedCollection new.
1 to: MetaRhos size do: [:m | trueRhos add: ((MetaRhos at: m)/ 100) asFloat].
max := trueRhos size.
rho < (trueRhos at: 1) ifTrue:

```

SmallTalk-80 Code For Class: Utils (continued)
--

```

[DialogView notify: 'This rho is too SMALL; The Minimum Rho is ',(trueRhos at: 1)
printString. ^self.].
rho > (trueRhos at: max) ifTrue:
[DialogView notify: 'This rho is too LARGE; The Maximum Rho is ',(trueRhos at: max)
printString. ^self.].
k := 1.
bracketed := false.
[bracketed] whileFalse: [
(rho >= (trueRhos at:k) and: [rho <= (trueRhos at: (k+1))])
ifTrue: [bracketed := true]
ifFalse: [k:=k+1].].
lo := trueRhos at:k.
loValues := MetaValues at: k.
hi := trueRhos at: (k+1).
hiValues := MetaValues at: (k+1).
interp := (rho-lo)/(hi-lo).
DialogView notify: ('Rho Value of ', rho printString, ' has been braceded by ',
((trueRhos at:k) printString), ' and ', ((trueRhos at:(k+1)) printString) ,
\Interpolation value is ', interp printString,
\ \ For This MetaModel to Perform as Expected',
\The Mean Part InterArrival Time Should be Set To ', (MetaService/rho) printString)
withCRs asComposedText centered.
MetaDistribution := SortedCollection new.

1 to:140 do: [:n |
lo := loValues at: n.
hi := hiValues at: n.
result := (interp * hi) + ((1-interp) * lo).
MetaDistribution add: result.].!

```

SmallTalk-80 Code For Class: WorkCenter

'From Objectworks(r)Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:35:45 pm'

```

Plant subclass: #WorkCenter
instanceVariableNames: 'resources name operator '
classVariableNames: "
poolDictionaries: "
category: 'Cim Controllers'

!WorkCenter methodsFor: 'task language'

produceOperator: anAmount of: aLabel
"Method to create an Operator and add it to the workcenter"

operator isNil
ifTrue: [^operator := Operator newWithName: aLabel andAmount: anAmount]
ifFalse: [self error: 'This needs to be checked out']!

operator
"Answer the operator belonging to this work center"

^operator!

```


SmallTalk-80 Code For Class: WorkCenter (continued)

```
printOperatorResultsOn: aStream
```

```
"Print the statistics of the operator output stream"
```

```
operator isNil ifFalse: [operator printResultsOn: aStream]!
```

```
!WorkCenter methodsFor: 'decisions'!
```

```
whatShouldIDo: aWFI
```

```
"This wfi needs a decision on what its next operation is"
```

```
"THIS IS A MODIFIED COPY TO IMPLEMENT A SPECIAL ROUTING RULE  
WFI WILL SEEK KIT2 UNLESS ITS QUEUE IS 4 OR MORE LONGER THAN KIT1  
IN WHICH CASE KIT1 WILL BE USED"
```

```
"THIS VERSION IS FOR USE IN THE QN2 MODEL - IT INCLUDES THE RANDOM  
ROUTER"
```

```
! queueLength operation q2 op2 q3 op3 rannum tester busy waiting !
```

```
queueLength := 987654.
```

```
aWFI areYouDone
```

```
    ifTrue: [self error: 'This wfi is done.']
```

```
    ifFalse: [(aWFI routing at: aWFI currentStage)
```

```
        do: [:op | "This is the list of current alternate operations"
```

```
            (self includesResourceFor: op machine)
```

```
                ifTrue:
```

```
                    ["Transcript cr; show: op machine , ' '; show:
```

```
resourceNamed: op machine) queueLength
```

```
name = 'qn2'
```

```
    ifTrue:
```

```
        [op machine = 'm2'
```

```
            ifTrue:
```

```
                [q2 :=
```

```
op2 :=
```

```
op machine = 'm3'
```

```
    ifTrue:
```

```
        [q3 :=
```

```
op3 :=
```

```
op machine = 'kit1'
```

```
    ifTrue:
```

```
        [q2 := queueLength := (self
```

```
op2 := op].
```

```
op machine = 'kit2'
```

```
    ifTrue:
```

```
        [q3 := queueLength := (self
```

```
op3 := op].
```

```
tester := (self resourceNamed: op machine)
```

```
(self
```

```
printString."
```

```
queueLength := (self resourceNamed: op machine) queueLength.
```

```
op].
```

```
queueLength := (self resourceNamed: op machine) queueLength.
```

```
op]].
```

```
resourceNamed: op machine) queueLength.
```

```
resourceNamed: op machine) queueLength.
```

```
queueLength.
```

SmallTalk-80 Code For Class: WorkCenter (continued)

```

[op machine = 'm4']]
machine) amountAvailable = 0 ifTrue: [busy := 1].
resourceNamed: op machine) waitingForInputqueueLength.

busy + waiting.

machine.

"Logic For PREFERRED QUEUE STRATEGY."
  name = 'wc1' ifTrue: [q3 - 4 < q2
    ifTrue: [operation := op3]
    ifFalse: [operation := op2]].
"Logic For SHORTEST QUEUE STRATEGY."
  name='wc1' ifTrue: [q3 >= q2
    ifTrue: [operation := op2]
    ifFalse: [operation := op3]].
name = 'qn2'
  ifTrue:
    [
      rannum := RandomRouter next.
      rannum < 0.5
        ifTrue: ["Transcript cr; show: 'RandomRouter ', rannum printString."
          operation := op2]
        ifFalse: [operation := op3]].
    ^operation"Transcript cr; show: 'Routed to: ', operation machine."!

whatShouldIDoV1: aWFI

"This wfi needs a decision on what its next operation is"
"THIS IS A COPY OF THE ORIGINAL METHOD"

lqueueLength operation l

queueLength := 987654.
aWFI areYouDone ifTrue: [self error: 'This wfi is done.']
ifFalse: [(aWFI routing at: aWFI currentStage) "This is the list of
  current alternate operations"
  do: [:opl
    (self includesResourceFor: op machine) ifTrue: [
      (((self resourceNamed: op machine) queueLength) < queueLength)
      ifTrue: [queueLength := (self resourceNamed: op machine) queueLength.
        operation := opl]
      ifFalse: [

```

busy := 0.
 waiting := 0.
 (op machine = 'm2' or: [op machine = 'm3' or:
 ifTrue: [(self resourceNamed: op
 waiting := (self
 tester + busy + waiting < queueLength
 ifTrue:
 [queueLength := tester +
 operation := opl]
 ifFalse: ["The next operation cannot be carried out at this
 Only the
 plant can decide this question"
 ^Plant active whatShouldIDo: aWFI]]].

"The next operation cannot be carried out at this machine. Only the

SmallTalk-80 Code For Class: WorkCenter (continued)

```
plant can decide this question"
^Plant active whatShouldIDo: aWFI]].
^operation!
```

SmallTalk-80 Code For Class: WorkFlowItem

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:36:41 pm'!

```
SimulationObject subclass: #WorkFlowItem
    instanceVariableNames: 'name entryTime currentLocation workStation queueEntryTime serial workCenter
currentStage currentOperation done routing dueDate hasOperator '
    classVariableNames: 'Count EntryTime MaterialHandling TimeInSystem '
    poolDictionaries: "
    category: 'Cim Sim!'
WorkFlowItem comment:
'Class WorkFlowItem represents the parts moving through the system which
must be processed by the system resources.'
```

```
!WorkFlowItem methodsFor: 'initialize-release'!
```

```
initialize
```

```
"The workflowitem starts from the buffer 'storage'"
```

```
currentStage := 1.
done := false.
hasOperator := 0.
super initialize!
```

```
!WorkFlowItem methodsFor: 'task language'!
```

```
acquireOperator
```

```
"acquire the workcenters operator if one exists"
"self halt."
"resourceNeeded := aResource.
amountNeeded := 1."
```

```
workCenter operator provideServiceTo: self.
hasOperator := 2.!
```

```
getProcessedAtLocation
```

```
"The wfi has arrived at a resource, and acquired it. Now complete the
current operation"
```

```
| resource time operTime suTime |
operTime := currentOperation processTime next.
suTime := currentOperation setUpTime next.
time := operTime + suTime.
resource := Plant active resourceNamed: currentOperation machine.
"SimScript cr; nextPutAll: self name , ' ', self serial printString , ' PT at ' , resource name , ' for Setup: ' ,
suTime printString , ' and Process: ' , operTime printString."
(workCenter name = 'ob2' or: [workCenter name = 'ob3' or:[workCenter name = 'ob4']])
```

SmallTalk-80 Code For Class: WorkflowItem (continued)

```

"(workCenter name = 'ob2' or: [workCenter name = 'ob3' ])"
  ifTrue:
    ["self halt."
     self acquireOperator.
     self holdFor: suTime.
     self releaseOperator.
     workCenter operator processTime: suTime].
  self holdFor: operTime.
  resource processTime: time!

hasOperator
  ^hasOperator!

hasOperator: aStatus
  hasOperator := aStatus.
  ^self.!

releaseOperator

" Release the operator when the resource is no longer required "

  workCenter operator releaseBy: self.
  hasOperator := 3.!

workgetProcessedAtLocation
  "The wfi has arrived at a resource, and acquired it. Now complete the
  current operation"

  | resource time operTime suTime |
  operTime := currentOperation processTime next.
  suTime := currentOperation setUpTime next.
  time := operTime + suTime.
  resource := Plant active resourceNamed: currentOperation machine.
  SimScript cr; nextPutAll: (self name), ',(self serial printString), ' PT at ', resource name, ' for
Setup: ',
    suTime printString, ' and Process: ', operTime printString.
  "SimScript cr; nextPutAll: self printString, ' obtained ', resource printString."
  ((workCenter name = 'ob2') or: [workCenter name='ob3']) ifTrue:[
    "self halt."
    self acquireOperator.
    self holdFor: suTime.
    self releaseOperator.
    workCenter operator processTime: suTime.
  ].
  self holdFor: operTime.
  resource processTime: time! !

```

SmallTalk-80 Code For Class: WorkStation
--

'From Objectworks(r)\Smalltalk, Release 4 of 25 February 1991 on 21 December 1991 at 4:36:08 pm'!

```

WorkCenter subclass: #WorkStation
  instanceVariableNames: 'wsAmountAvailable wsQueueController wsProcessingTimes wsUtilization
waitingForInputQ blockingWFI blocked workCenter failureDistribution repairDistribution '

```

SmallTalk-80 Code For Class: WorkStation (continued)
--

```

classVariableNames: "
poolDictionaries: "
category: 'Cim Resources'!

WorkStation comment:
'Class WorkStation is the class which represents a delayer to an object
being processed. This class and its subclasses are used to represent
machine resources in the system.!'

!WorkStation methodsFor: 'task language'!

provideServices
    "provide workstation resources to the next job in queue"
    |waitingWFI wfi |

    [wsQueueController inputQueueEmpty not and: [wfi := wsQueueController next. wfi amountNeeded <=
wsAmountAvailable]]
    whileTrue:
        [waitingWFI := wsQueueController inputQueueRemove: wfi.
wsAmountAvailable := wsAmountAvailable - waitingWFI amountNeeded.
self name = 'meta' ifTrue:[wsAmountAvailable := wsAmountAvailable + waitingWFI amountNeeded].
[self hasInputSpace2 and: [waitingForInputQ isEmpty not]] whileTrue:
    [self reserveAPlace. waitingForInputQ removeFirst resume].
wsUtilization equals: (wsUtilization value + waitingWFI amountNeeded).
waitingWFI resume.].
"self name = 'meta' ifTrue: [self halt]."!

provideServiceTo: aWFI
    "This wfi needs to be serviced. Put into the queue, and provide
a server if possible"

    wsQueueController addToInputQueue: aWFI.
    "SimScript cr; nextPutAll: aWFI printString , ' needs: ', self printString , ' at ', Simulation active time
printString."
    self provideServices.
    aWFI pause.
    aWFI hasOperator: 1.
    "SimScript cr; nextPutAll: aWFI printString , ' grabbed: ', self printString , ' at ', Simulation active time
printString."!

!WorkStation methodsFor: 'initialize-release'!

failureDistribution: aDistribution repairDistribution: another
    failureDistribution := aDistribution.
    repairDistribution := another.
    name = 'm1'
        ifTrue:
            [repairDistribution random seed: (TheSeedArray at: 14).
failureDistribution random seed: (TheSeedArray at: 15)].
    name = 'm2'
        ifTrue:
            [repairDistribution random seed: (TheSeedArray at: 16).
failureDistribution random seed: (TheSeedArray at: 17)].
    name = 'm3'
        ifTrue:

```

SmallTalk-80 Code For Class: WorkStation (continued)
--

```

[repairDistribution random seed: (TheSeedArray at: 18).
failureDistribution random seed: (TheSeedArray at: 19)].

name = 'm4'
  ifTrue:
    [repairDistribution random seed: (TheSeedArray at: 20).
failureDistribution random seed: (TheSeedArray at: 21)].

"self halt."
self scheduleFailureRepairCycle.!

failYourselfFor: aDownTime
| dummy t1 t2 check |
"SimScript cr;nextPutAll: self printString , ' failed!! ', 'downTime = ', aDownTime printString ,
' at ', Simulation active time printString."
wsAmountAvailable = 1
  ifTrue: ["SimScript cr; nextPutAll: self printString , ' was idle - FAILURE IGNORED"]
  ifFalse:
    ["SimScript cr; nextPutAll: self printString , ' was busy '."
check := Simulation active postponeEventForResource: self by: aDownTime.
"SimScript cr; nextPutAll: 'check class: ', check class printString."
check isNil
  ifTrue:
    [wsUtilization equals: 0.
dummy := (WorkFlowItem new initialize) name: 'Repair WFI';

workCenter: workCenter; location: self.

t1 := Simulation active time.
dummy acquireOperator.
t2 := Simulation active time.
self holdFor: aDownTime.
dummy releaseOperator.
dummy := nil.
Simulation active postponeEventForResource2: self by: t2 - t1 +

aDownTime - 100.

wsUtilization equals: 1]]!

workfailYourselfFor: aDownTime
| dummy t1 t2 check |
SimScript cr; nextPutAll: self printString , ' failed!! ', 'downTime = ', aDownTime printString , ' at ',
Simulation active time printString.

wsAmountAvailable = 1
  ifTrue:
    ["wsAmountAvailable := wsAmountAvailable - 1."
SimScript cr; nextPutAll: self printString , ' was idle - Failure IGNORED'.
"Simulation active delayFor: aDownTime.
SimScript cr; nextPutAll: self printString , ' is up now!! at ', Simulation active time

printStats.

wsAmountAvailable := wsAmountAvailable + 1.
self provideServices"]
  ifFalse:
    [SimScript cr; nextPutAll: self printString , ' was busy '.
check := Simulation active postponeEventForResource: self by: aDownTime.
check isNil ifTrue:[
SimScript cr; nextPutAll: 'check class: ', check class printString.
wsUtilization equals: 0.

```

SmallTalk-80 Code For Class: WorkStation (continued)
--

```

dummy := WorkflowItem new initialize name: 'Repair WFI'; workCenter: workCenter;
location: self.

t1 := Simulation active time.
dummy acquireOperator.
t2 := Simulation active time.
self holdFor: aDownTime.
dummy releaseOperator.
dummy := nil.
Simulation active postponeEventForResource2: self by: (t2-t1+aDownTime-100).
wsUtilization equals: 1]]! !

!WorkStation methodsFor: 'queue capacity'!

putMeInOutputQueue: aWFI

"output queues are adjacent to the workstation,
and do not need reservation.
WFI's directly move into them, without worrying about other
competition, if there is a space. If there is no space, they are blocked"

wsQueueController outputHasSpace ifTrue: [
wsQueueController putInOutput: aWFI]
ifFalse: [ "There is no place for this wfi. the workStation is blocked"
"self halt."
"SimScript cr; nextPutAll: aWFI printString, ' blocked the ', self printString, ' at ', Simulation active time
printString."
BlockCount := BlockCount + 1.
blocked := true.
blockingWFI := aWFI.
"self halt."
aWFI pause.
"SimScript cr; nextPutAll: aWFI printString, ' unBlocked the ', self printString, ' at ', Simulation active
time printString."
blocked := false.
wsQueueController putInOutput: aWFI!]

!WorkStation methodsFor: 'testing'!

hasInputSpace2

"There is space in the workStation if the input queue has space
or if there is a server available"
"SimScript cr; nextPutAll: 'Checking for Input Space (provideServices) at ', self name, ' at ',
Simulation active time printString, ' Answer: ', (wsQueueController inputHasSpace: wsAmountAvailable)
printString."
"self halt."
^wsQueueController inputHasSpace: wsAmountAvailable!

```

APPENDIX B

SIMULATION RUN DESIGN CONSIDERATIONS

As stated in Chapter VIII, the simulation run design considerations are designed to answer the following questions:

- o How is "to approximately model" to be judged?
- o How long must each simulation run be to ensure that the time-in-system distributions are approximately modeled?
- o How long a "warm-up" period should be allowed to eliminate the idle and empty start-up influence on the collected time-in-system statistics?
- o How many simulation runs at each utilization value are required to ensure that the time-in-system distributions are approximately modeled?
- o How should random deviates be generated for each simulation run?

Each of these questions will be addressed in turn in the sections below.

How Does One Judge "Approximately Modeled"?

Within the context of this research, the stated objective of a metamodel is to approximately preserve the time-in-system distribution for parts moving through the workcenter. In light of this, the question of "approximately modeled" becomes one of judging whether the time-in-system distribution generated from the metamodel exhibits a "goodness-of-fit" when compared to the corresponding base model distribution.

The null hypothesis for virtually all goodness of fit tests is that the observed values are independent identically distributed random variables with distribution function F^* . A cautionary note is appropriate regarding the testing of this hypothesis [Law and Kelton 1991, sec. 6.6.2]. Failure to reject the null hypothesis should not be interpreted as accepting it to be true. For small numbers of observed values, the tests should be viewed as a systematic approach to detect gross differences. For large numbers of observed values, the tests almost always reject the null hypothesis since it is rarely exactly true. This is unfortunate since in most cases, what is needed is a distribution that is approximately correct.

Perhaps the two most popular forms of general goodness of fit tests are the chi-square test [Law and Kelton 1991, 382-387] and the Kolmogorov-Smirnov (K-S) test [Massey 1951]. For a continuous random variable, the chi-square test can be thought of as a comparison of the observed and hypothesized probability mass functions. The K-S test can be thought of as a comparison of the cumulative distribution functions. For this research, the K-S test was favored over the chi-squared test for the following two reasons:

- o The K-S test tends to be more powerful against many alternatives [Stephens 1974].
- o The K-S test does not require the selection of equiprobable intervals each containing at least five observations.

The K-S test employed in comparing metamodels and base models in this research must be deemed an approximate test for the following reasons:

- o The test was applied to grouped data (see Appendix E) rather than to individual values. Massey [1951] indicates that this grouping tends to lower the significance levels of the test.
- o The test was applied using a sample of 3,000 averaged observations calculated from 15,000 individual observations grouped into 140 cells. The "n" factor in the K-S test was set to 3,000. Using 15,000 would have lowered significance levels; 140 would have raised them.

While the test must be considered approximate, this fact in itself does not affect the validity of the developed methodology. The metamodel validation test (of which the K-S test is a part) is one component of a creation-validation-remedial cycle of procedures designed to conclude with a valid metamodel. Any goodness of fit test could be applied in the validation stage. A more powerful test would result in additional remedial cycles, a less powerful test in fewer. The final decision on the exact test to be used and its parameters is a function of how "approximate" the metamodel behavior is allowed to

become. Within this research effort, the approximate K-S test described above and detailed in Massey [1951] appears both reasonable and satisfactory.

How Long Should Each Simulation Run Be?

and

How Many Simulation Runs (Repetitions) Are Required For Scenario At Each ρ Value?

The length of each simulation run and the number of independent runs (repetitions) were the subject of considerable empirical investigation at the inception of this research. The two issues were considered simultaneously due to their interrelationship. The interrelationship is in the form of a tradeoff, more shorter runs versus fewer longer runs.

The empirical investigation was centered around the performance of the two queueing network metamodels QN1 and QN2. Both of these metamodels had known closed form solutions. By graphically comparing the averaged cumulative distribution function obtained from the simulation runs against the known analytical solution, it was possible to visually assess the degree to which the two coincided. In addition to this pragmatic visual comparison, the K-S test described in the previous section was used as an objective measure of performance.

The process of finalizing the run length and number of runs involved testing various values at several utilization levels (ρ) across several models (QN1 and QN2). The run lengths tested ranged from a low of 1,000 observations of time-in-system to a high of 15,000 observations. The number of runs (repetitions) ranged from a low of three to a high of ten. While run lengths of 15,000 observations produced excellent results (within even stringent K-S limits), the run times were excessive (> 45 minutes per run). Similarly, the average based on ten repetitions showed excellent results but the total run times were excessive. The final values used for the research, five repetitions each with 3,000 observations, were selected as a reasonable and quite acceptable compromise. Undoubtedly many other combinations could have delivered acceptable results. This pair

simply surfaced first and withstood the pragmatic testing as being both rational and practical.

How Long Should The Simulation "Warm-Up" Period Be?

The warm up period for a simulation run is that period of time required for the simulation to reach steady state behavior. It allows the system time to mitigate the effects of starting empty and idle. Detecting the end of warm-up and the start of steady state behavior is more of an art than a science. One of the most common techniques used is to view a time ordered graph of a relevant system performance measure (or its moving average) and eliminate the "ramp-up" effect typical of non-steady-state behavior. This was the basic approach used throughout this research.

Graphs of time-in-system produced during initial simulation runs indicated that the empty and idle effect was mitigated in less than 100 time units. For complex plant configurations and higher utilization factors (ρ 's), the warm up period was approximately 75 time units. For less complex plants and low ρ 's, the warm up period was frequently less than 25 time units.

Based on these results and incorporating a threefold safety factor, the warm-up time for all simulation runs was set at 300 time units. When simulated time reached 300 in each run, the statistical arrays were cleared and the count for the 3,000 time-in-system observations was initiated. At the conclusion of each run, a time-ordered graph of the collected time-in-system observations was displayed for review. In no case during the conduct of this research did this end-of-run time-ordered graph display a ramp-up effect.

How Should Random Deviates Be Generated?

The problem of generating "good" random numbers and random deviates has long plagued the simulation community (see for instance [Park and Miller 1988]). This research was no different. The questions surfaced in the empirical investigation to

determine run lengths and number of repetitions (see section above). Initial results in this investigation showed that both long run lengths (>8000) and high repetition counts (>5) were going to be needed to successfully pass the visual and K-S tests.

Fortunately, from the author's perspective, two researchers at The University of Oklahoma, Steve Tretheway and Mike Oltmanns, were actively investigating this area. The most significant results of their (unpublished) efforts were:

- o Park and Miller's [1988] random number generator was implemented;
- o Ahrens and Dieter's [1974] gamma deviate generator ($\alpha < 1$) was implemented;
- o Cheng's [1977] gamma deviate generator ($\alpha \geq 1$) was implemented.

While this provided some improvement in the run lengths and number of repetitions required to achieve acceptable results, more improvement was desirable.

One final improvement in the random deviate generation process led to the results which were ultimately implemented within this research. That improvement was the implementation of multiple stream random number generation. At the suggestion of committee chair J.H. Mize and committee member M. Kamath, an independent random number generator was implemented for each stochastic process in the simulation model. Previous results (e.g., Mize [1973]) had demonstrated the superiority of this approach over single stream random number generation. This work further substantiates that benefit. The final result after implementation of all the above improvements was that acceptable performance of metamodels working against known analytical solutions was achieved using 5 repetitions with 3,000 observations each.

APPENDIX C
ANALYSIS OF VARIANCE OF EXPERIMENTAL
DATA

PLANT LEVEL VALIDATION #2
STATISTICAL RATIONALE

General Description:

Plant level validation #2 measures the effect of implementing a "decision" within the plant. The decision impacts (possibly) the average distribution of time-in-system for parts moving through the plant. The statistical analysis given below is designed to estimate the change in mean time-in-system under two different decisions and determine the observed significance level (OSL) of a test of no difference.

Factors:

Decision (D) - a fixed factor with d=2 levels;

Random Number Set (R) - a random factor with r=5 levels;

Number of Observations per Cell - n=1.

Model:

The data are in a two-way cross classification. Assuming that normal theory assumptions hold, a model for the mean time-in-system statistic is:

$$MTIS_{ij} = \mu + D_i + R_j + DR_{ij} + \epsilon_{ij}$$

where:

$MTIS_{ij}$ = mean time-in-system with Decision i and Random Number Set j;

μ = common effect for the whole experiment;

D_i = effect of the i-th Decision;

R_j = effect of the j-th Random Number Set;

DR_{ij} = interaction effect of the i-th Decision and the j-th Random Number Set;

ϵ_{ij} = random effect.

Expected Mean Squares (general)

The general expressions for the expected mean squares for this experimental model are developed in the table below using the methodology of Hicks [1964, p. 153].

Sources of Variation	Degrees of Freedom	Expected Mean Squares
Total	$drn-1$	
Decision Level (D)	$d-1$	$\sigma_e^2 + n\sigma_{DR}^2 + m \frac{\sum D_i^2}{d-1}$
Random Number Set (R)	$r-1$	$\sigma_e^2 + dn\sigma_R^2$
Interaction (DR)	$(d-1)(r-1)$	$\sigma_e^2 + n\sigma_{DR}^2$
Random (ϵ)	$dr(n-1)$	σ_e^2

Expected Mean Squares (specific)

After substituting specific values, the expressions for the expected mean squares of the model for the data are shown below.

Sources of Variation	Degrees of Freedom	Expected Mean Squares
Total	9	
Decision(D)	1	$\sigma_e^2 + \sigma_{DR}^2 + 5 \sum D_i^2$
Random Number Set (R)	4	$\sigma_e^2 + 2\sigma_R^2$
Interaction (DR)	4	$\sigma_e^2 + \sigma_{DR}^2$
Random (ϵ)	0	σ_e^2

Hypothesis:

The hypothesis to be tested is whether the variation due to the decision is zero. In terms of the above model and expected mean squares this becomes:

$H_0: D_i = 0$ for all i ;

H_1 : at least one $D_i \neq 0$.

F Test:

Based upon the expected mean squares (EMS) and assuming that normal theory assumptions hold, the appropriate F-statistic to test for significance of the hypothesis given above is formed by:

$$F_{\text{calc}} = \frac{\text{EMS}(D)}{\text{EMS}(DR)}$$

Conclusions:

The decision resulting from the analysis is based upon the observed significance level (OSL) of F_{calc} . The OSL is a measure of the evidence against H_0 . The smaller it is the larger the evidence since it is computed assuming H_0 is true. If the value is small, it means that the data is rare given that H_0 is true. The decision must be made (based on the evidence) that either: (1) H_0 is true and a rare event has occurred, or (2) H_0 is false.

For a specified α level:

If $\text{OSL} \leq \alpha$; then reject H_0 ;

otherwise, fail to reject H_0 .

In terms of the current research, if H_0 is rejected, we conclude that a statistically significant difference exists between the mean time-in-system of the two levels of the

decision variable (i.e., fast vs slow inspection station speed or shortest vs preferred queue selection strategy).

PLANT LEVEL VALIDATION #2

SAS PROGRAM

```

/*
Dissertation Plant Level Validation #2
ANOVA for Differences in Treatment Means
*/

***** Set general parameters;
options ps=58 nodate nonumber;
%let plant = QN2;
libname dbp "c:\dbp\&plant";
title1 "&plant Plant Level Validation";
title2 'Time In System - Validation #2 - Differences in Treatment Means';

***** Read raw data;
data raw; set dbp.avg&plant;run;

*=====;
***** Test for BASE model, 0.500 Rho, SHORT queues, SLOW vs FAST Inspection;
title3 'Base Model - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection';

***** Select and Print Data;
data test1; set raw;
  if (model='BASE' and rho='0.500' and insp='SLOW' and que='SHORT')
    or (model='BASE' and rho='0.500' and insp='FAST' and que='SHORT');
run;
proc print data=test1; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test1;
  class insp run;
  model mean = insp run insp*run;
  test h=insp e=insp*run / etype=1 htype=1;
quit;

*=====;
***** Test for META model, 0.500 Rho, SHORT queues, SLOW vs FAST Inspection;
title3 'Meta Model - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection';

***** Select and Print Data;
data test2; set raw;
  if (model='META' and rho='0.500' and insp='SLOW' and que='SHORT')
    or (model='META' and rho='0.500' and insp='FAST' and que='SHORT');
run;

```

```

proc print data=test2; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test2;
  class insp run;
  model mean = insp run insp*run;
  test h=insp e=insp*run / etype=1 htype=1;
quit;

*=====;
***** Test for BASE model, 0.675 Rho, SHORT queues, SLOW vs FAST Inspection;
title3 'Base Model - Rho 0.675 - Shortest Queue - Fast vs Slow Inspection';

***** Select and Print Data;
data test3; set raw;
  if (model='BASE' and rho='0.675' and insp='SLOW' and que='SHORT')
    or (model='BASE' and rho='0.675' and insp='FAST' and que='SHORT');
run;
proc print data=test3; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test3;
  class insp run;
  model mean = insp run insp*run;
  test h=insp e=insp*run / etype=1 htype=1;
quit;

*=====;
***** Test for META model, 0.675 Rho, SHORT queues, SLOW vs FAST Inspection;
title3 'Meta Model - Rho 0.675 - Shortest Queue - Fast vs Slow Inspection';

***** Select and Print Data;
data test4; set raw;
  if (model='META' and rho='0.675' and insp='SLOW' and que='SHORT')
    or (model='META' and rho='0.675' and insp='FAST' and que='SHORT');
run;
proc print data=test4; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test4;
  class insp run;
  model mean = insp run insp*run;
  test h=insp e=insp*run / etype=1 htype=1;
quit;

*=====;

```

```
***** Test for BASE model, 0.500 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 'Base Model - Rho 0.500 - Fast Inspection - Shortest vs Preferred Queues';
```

```
***** Select and Print Data;
```

```
data test5; set raw;
  if (model='BASE' and rho='0.500' and insp='FAST' and que='SHORT')
    or (model='BASE' and rho='0.500' and insp='FAST' and que='PREF ');
run;
proc print data=test5; run;
```

```
***** Proc GLM to produce ANOVA and F-statistic;
```

```
proc glm data=test5;
  class que run;
  model mean = que run que*run;
  test h=que e=que*run / etype=1 htype=1;
quit;
```

```
*=====;
```

```
***** Test for META model, 0.500 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 'Meta Model - Rho 0.500 - Fast Inspection - Shortest vs Preferred Queues';
```

```
***** Select and Print Data;
```

```
data test6; set raw;
  if (model='META' and rho='0.500' and insp='FAST' and que='SHORT')
    or (model='META' and rho='0.500' and insp='FAST' and que='PREF ');
run;
proc print data=test6; run;
```

```
***** Proc GLM to produce ANOVA and F-statistic;
```

```
proc glm data=test6;
  class que run;
  model mean = que run que*run;
  test h=que e=que*run / etype=1 htype=1;
quit;
```

```
*=====;
```

```
***** Test for BASE model, 0.675 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 'Base Model - Rho 0.675 - Fast Inspection - Shortest vs Preferred Queues';
```

```
***** Select and Print Data;
```

```
data test7; set raw;
  if (model='BASE' and rho='0.675' and insp='FAST' and que='SHORT')
    or (model='BASE' and rho='0.675' and insp='FAST' and que='PREF ');
run;
proc print data=test7; run;
```

```
***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test7;
  class que run;
  model mean = que run que*run;
  test h=que e=que*run / etype=1 htype=1;
quit;

*=====;
***** Test for META model, 0.675 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 'Meta Model - Rho 0.675 - Fast Inspection - Shortest vs Preferred Queues';

***** Select and Print Data;
data test8; set raw;
  if (model='META' and rho='0.675' and insp='FAST' and que='SHORT')
    or (model='META' and rho='0.675' and insp='FAST' and que='PREF ');
run;
proc print data=test8; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test8;
  class que run;
  model mean = que run que*run;
  test h=que e=que*run / etype=1 htype=1;
quit;
```

PLANT LEVEL VALIDATION #2

SAMPLE SAS OUTPUT

OB4 Plant Level Validation
 Time In System - Validation #2 - Differences in Treatment Means
 Base Model - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection

OBS	RUN	MEAN	MODEL	RHO	INSP	QUE
1	B	12.1060	BASE	0.500	FAST	SHORT
2	D	12.0253	BASE	0.500	FAST	SHORT
3	J	12.1194	BASE	0.500	FAST	SHORT
4	K	12.0232	BASE	0.500	FAST	SHORT
5	S	12.1184	BASE	0.500	FAST	SHORT
6	B	12.4857	BASE	0.500	SLOW	SHORT
7	D	12.3729	BASE	0.500	SLOW	SHORT
8	J	12.4952	BASE	0.500	SLOW	SHORT
9	K	12.3752	BASE	0.500	SLOW	SHORT
10	S	12.4835	BASE	0.500	SLOW	SHORT

OB4 Plant Level Validation
 Time In System - Validation #2 - Differences in Treatment Means
 Base Model - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection

General Linear Models Procedure
 Class Level Information

Class	Levels	Values
INSP	2	FAST SLOW
RUN	5	B D J K S

Number of observations in data set = 10

General Linear Models Procedure

Dependent Variable: MEAN

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	0.35694254	0.03966028	.	.
Error	0
Corrected Total	9	0.35694254			

R-Square	C.V.	Root MSE	MEAN Mean
1.000000	0	0	12.2604896

Source	DF	Type I SS	Mean Square	F Value	Pr > F
INSP	1	0.33135237	0.33135237	.	.
RUN	4	0.02518839	0.00629710	.	.
INSP*RUN	4	0.00040178	0.00010044	.	.

Source	DF	Type III SS	Mean Square	F Value	Pr > F
INSP	1	0.33135237	0.33135237	.	.
RUN	4	0.02518839	0.00629710	.	.
INSP*RUN	4	0.00040178	0.00010044	.	.

Tests of Hypotheses using the Type I MS for INSP*RUN as an error term

Source	DF	Type I SS	Mean Square	F Value	Pr > F
INSP	1	0.33135237	0.33135237	3298.88	0.0001

PLANT LEVEL VALIDATION #3
STATISTICAL RATIONALE

General Description:

Plant level validation #3 compares the effect of implementing a "decision" within the plant using two different models - the base model and the meta model. The decision impacts (possibly) the mean time-in-system for parts moving through the plant. The statistical analysis below measures the observed significance level (OSL) of the difference between the effect in the base model and the effect in the metamodel.

Factors:

Model (M) - a fixed factor with m=2 levels;

Decision (D) - a fixed factor with d=2 levels;

Random Number Set (R) - a random factor with r=5 levels;

Number of Observations per Cell - n=1.

Model:

The data are in a three-way nested cross-classification. Assuming that normal theory assumptions hold, a model for the statistic described is given by¹:

$$MTIS_{ijk} = \mu + M_i + D_j + MD_{ij} + R_{k(i)} + DR_{jk(i)} + \epsilon_{ij}$$

where:

$MTIS_{ij}$ = mean time-in-system with Model i, Decision j, and Random Number Set k;

μ = common effect for the whole experiment;

M_i = effect of the i-th Model;

D_j = effect of the j-th Decision;

¹ A special thanks goes to Dr. David Weeks of the OSU Statistics Department for his assistance in formulating this model and assisting with interpretation of results.

MD_{ij} = the interaction effect of the i -th Model and the j -th Decision (measures the failure of the decision differences to be the same for both models);

$R_{k(i)}$ = the effect of the k -th Random Number Set within the i -th Model (this effect contains the confounded effects of the Random Number Set and the Random Number Set X Model interaction);

$DR_{jk(i)}$ = the interaction effect of the j -th Decision and the k -th Random Number Set within the i -th Model (This effect contains the confounded effects of the Decision X Random Number Set interaction and the Decision X Random Number Set X Model interaction. It measures the failure of decision differences to be the same over the runs inside each model and run, averaged);

ϵ_{ijk} = random effect.

Expected Mean Squares (general)

The general expressions for the expected mean squares for this experimental model are developed in the table below using the methodology of Hicks [1964, p. 153].

Sources of Variation	Degrees of Freedom	Expected Mean Squares
Total	$mdrn-1$	
Model (M)	$m-1$	$\sigma_e^2 + dn\sigma_R^2 + drn \frac{\sum M_i^2}{m-1}$
Decision (D)	$d-1$	$\sigma_e^2 + n \frac{d}{d-1} \sigma_{DR}^2 + mnrn \frac{\sum D_i^2}{d-1}$
Model x Decision Interaction (MD)	$(d-1)(m-1)$	$\sigma_e^2 + n \frac{d}{d-1} \sigma_{DR}^2 + nrn \frac{\sum (MD)_{ij}^2}{(m-1)(d-1)}$
Random Number Set Within Model (R)	$m(r-1)$	$\sigma_e^2 + dn\sigma_R^2$
Decision x Random Number Set Interaction Within Model (DR)	$m(d-1)(r-1)$	$\sigma_e^2 + n \frac{d}{d-1} \sigma_{DR}^2$
Random (ϵ)	$dr(n-1)$	σ_e^2

Expected Mean Squares (specific)

After substituting known values, the specific expressions for the expected mean squares for this experimental model are shown in the table below.

Sources of Variation	Degrees of Freedom	Expected Mean Squares
Total	19	
Model (M)	1	$\sigma_e^2 + 2\sigma_R^2 + 10 \sum M_i^2$
Decision (D)	1	$\sigma_e^2 + 2\sigma_{DR}^2 + 10 \sum D_i^2$
Model x Decision Interaction (MD)	1	$\sigma_e^2 + 2\sigma_{DR}^2 + 5 \sum (MD)_{ij}^2$
Random Number Set Within Model (R)	8	$\sigma_e^2 + 2\sigma_R^2$
Decision x Random Number Set Interaction Within Model (DR)	8	$\sigma_e^2 + 2\sigma_{DR}^2$
Random (ϵ)	0	σ_e^2

Hypothesis:

The hypothesis to be tested is whether the variation due to the interaction between models and decisions is zero. In terms of the above model and expected mean squares this becomes:

$$H_0: (MD)_{ij} = 0 \text{ for all } i \text{ and } j;$$

$$H_1: \text{at least one } MD_{ij} \neq 0.$$

F Test:

Based upon the expected mean squares (EMS) the appropriate F-statistic to test for significance of the interaction between model and decision is formed by:

$$F_{\text{calc}} = \frac{\text{EMS}(\text{MD})}{\text{EMS}(\text{DR})}$$

Conclusions:

The decision resulting from the analysis is based upon the observed significance level (OSL) of F_{calc} . For a specified α level:

If $\text{OSL} \leq \alpha$; then reject H_0 ;

otherwise; fail to reject H_0 .

In terms of the current research, if H_0 is rejected, we conclude that a significant difference exists between the average behavior of the base model and the average behavior of the metamodel with respect to changes in the decision factor. Interpreted graphically, this implies that a significant difference exists between the slopes of the two lines shown on the validation #1 (visual inspection) graphs.

PLANT LEVEL VALIDATION #3

SAS PROGRAM

```

/*
Dissertation Plant Level Validation #3
ANOVA for Difference in Treatment Slopes across Models
*/

***** Set general parameters;
options ps=58 nodate nonumber;
libname dbp "c:\dbp\&plant.";
%let plant = QN2;
title1 "&plant. Plant Level Validation";
title2 "Mean Time In System - Validation #3 - Differences in Treatment Slopes";

***** Read raw data;
data raw; set dbp.avg&plant.; run;

*=====;
***** Test for 0.500 Rho, SHORT queues, SLOW vs FAST inspection;
title3 "&plant. - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection";

***** Select and Print Data;
data test1; set raw;
  if (rho="0.500" and insp="SLOW" and que="SHORT")
  or (rho="0.500" and insp="FAST" and que="SHORT");
run;
proc print data=test1; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test1;
  class model insp run;
  model mean = model run(model)
  insp insp*model insp*run(model) ;
  test h=model e=run(model) / etype=1 htype=1;
  test h=insp insp*model e=insp*run(model) / etype=1 htype=1;
quit;

*=====;
***** Test for 0.675 Rho, SHORT queues, SLOW vs FAST inspection;
title3 "&plant. - Rho 0.675 - Shortest Queue - Fast vs Slow Inspection";

***** Select and Print Data;
data test1; set raw;
  if (rho="0.675" and insp="SLOW" and que="SHORT")

```

```

    or (rho="0.675" and insp="FAST" and que="SHORT");
run;
proc print data=test1; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test1;
  class model insp run;
  model mean = model run(model)
    insp insp*model insp*run(model) ;
  test h=model e=run(model) / etype=1 htype=1;
  test h=insp insp*model e=insp*run(model) / etype=1 htype=1;
quit;

*=====;
***** Test for 0.500 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 "&plant. - Rho 0.500 - Fast Inspection - Short vs Preferred Queues";

***** Select and Print Data;
data test1; set raw;
  if (rho="0.500" and insp="FAST" and que="SHORT")
    or (rho="0.500" and insp="FAST" and que="PREF ");
run;
proc print data=test1; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test1;
  class model que run;
  model mean = model run(model)
    que que*model que*run(model) ;
  test h=model e=run(model) / etype=1 htype=1;
  test h=que que*model e=que*run(model) / etype=1 htype=1;
quit;

*=====;
***** Test for 0.675 Rho, FAST insp., SHORT vs PREFERRED queues;
title3 "&plant. - Rho 0.675 - Fast Inspection - Short vs Preferred Queues";

***** Select and Print Data;
data test1; set raw;
  if (rho="0.675" and insp="FAST" and que="SHORT")
    or (rho="0.675" and insp="FAST" and que="PREF ");
run;
proc print data=test1; run;

***** Proc GLM to produce ANOVA and F-statistic;
proc glm data=test1;

```

```
class model que run;  
model mean = model run(model)  
  que que*model que*run(model) ;  
test h=model e=run(model) / etype=1 htype=1;  
test h=que que*model e=que*run(model) / etype=1 htype=1;  
quit;
```

PLANT LEVEL VALIDATION #3

SAMPLE SAS OUTPUT

OB4 Plant Level Validation
 Mean Time In System - Validation #3 - Differences in Treatment Slopes
 OB4 - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection

OBS	RUN	MEAN	MODEL	RHO	INSP	QUE
1	B	12.1060	BASE	0.500	FAST	SHORT
2	D	12.0253	BASE	0.500	FAST	SHORT
3	J	12.1194	BASE	0.500	FAST	SHORT
4	K	12.0232	BASE	0.500	FAST	SHORT
5	S	12.1184	BASE	0.500	FAST	SHORT
6	B	12.7432	META	0.500	FAST	SHORT
7	D	12.6981	META	0.500	FAST	SHORT
8	J	12.7809	META	0.500	FAST	SHORT
9	K	12.7021	META	0.500	FAST	SHORT
10	S	12.7652	META	0.500	FAST	SHORT
11	B	12.4857	BASE	0.500	SLOW	SHORT
12	D	12.3729	BASE	0.500	SLOW	SHORT
13	J	12.4952	BASE	0.500	SLOW	SHORT
14	K	12.3752	BASE	0.500	SLOW	SHORT
15	S	12.4835	BASE	0.500	SLOW	SHORT
16	B	13.2108	META	0.500	SLOW	SHORT
17	D	13.1558	META	0.500	SLOW	SHORT
18	J	13.2422	META	0.500	SLOW	SHORT
19	K	13.1276	META	0.500	SLOW	SHORT
20	S	13.2426	META	0.500	SLOW	SHORT

General Linear Models Procedure
 Class Level Information

Class	Levels	Values
MODEL	2	BASE META
INSP	2	FAST SLOW
RUN	5	B D J K S

Number of observations in data set = 20

General Linear Models Procedure

Dependent Variable: MEAN

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	19	3.39217824	0.17853570		
Error	0				
Corrected Total	19	3.39217824			

R-Square	C.V.	Root MSE	MEAN Mean
1.000000	0	0	12.6136694

OB4 Plant Level Validation
 Mean Time In System - Validation #3 - Differences in Treatment Slopes
 OB4 - Rho 0.500 - Shortest Queue - Fast vs Slow Inspection

Source	DF	Type I SS	Mean Square	F Value	Pr > F
MODEL	1	2.49471875	2.49471875	.	.
RUN(MODEL)	8	0.04072875	0.00509109	.	.
INSP	1	0.84454999	0.84454999	.	.
MODEL*INSP	1	0.01100972	0.01100972	.	.
INSP*RUN(MODEL)	8	0.00117102	0.00014638	.	.

Source	DF	Type III SS	Mean Square	F Value	Pr > F
MODEL	1	2.49471875	2.49471875	.	.
RUN(MODEL)	8	0.04072875	0.00509109	.	.
INSP	1	0.84454999	0.84454999	.	.
MODEL*INSP	1	0.01100972	0.01100972	.	.
INSP*RUN(MODEL)	8	0.00117102	0.00014638	.	.

Tests of Hypotheses using the Type I MS for RUN(MODEL) as an error term

Source	DF	Type I SS	Mean Square	F Value	Pr > F
MODEL	1	2.49471875	2.49471875	490.02	0.0001

Tests of Hypotheses using the Type I MS for INSP*RUN(MODEL) as an error term

Source	DF	Type I SS	Mean Square	F Value	Pr > F
INSP	1	0.84454999	0.84454999	5769.67	0.0001
MODEL*INSP	1	0.01100972	0.01100972	75.21	0.0001

APPENDIX D
PERSPECTIVES ON COMPARING TWO DERIVED
NUMBERS

PERSPECTIVES ON COMPARING TWO DERIVED NUMBERS

Frequently the question is asked "are these two derived numbers the same or different". To truly answer this question, one must understand the perspective from which the question is asked. At least three perspectives are clearly distinguishable: a numerical perspective, a statistical perspective, and a practical perspective. In its purest sense, a numerical comparison of numbers will rarely yield the conclusion that two numbers are equal. If expanded to enough decimal places almost any two real numbers will eventually differ. Usually, this perspective is not the one from which the comparison question is asked.

The statistical perspective is based on sampling theory and the laws of probability. The basic approach says that if you know (1) how big a difference you are interested in detecting, (2) how much natural variation is present in the process, and (3) how much risk of error you are willing to accept, then you can determine how many data points you need to collect to determine if two outcomes are the same or different. The answer to the comparison question hinges on the particular experimental design used to collect data to answer it.

Within the context of this research effort, the important relationship in the statistical perspective is the relationship between the sample size (n) and the size of the difference detectable (δ). The relationship is an inverse square relationship of the form: $\delta \sim 1/\sqrt{n}$ [Hicks 1964, sec. 2.5; Steel and Torrie 1980, sec. 5.12]. During the simulation run design phase of the research (see Appendix B), it is empirically determined that a sample size (n) of 3,000 is needed to adequately capture the "tail behavior" of the time-in-system distribution. By virtue of this large sample size, very small differences (δ) in the mean time-in-system for different options become "significantly" different. The analysis of variance results presented in Chapter VIII reflect this effect. Under these circumstances, the question that must be asked is whether or not these statistically

significant differences represent a "practical" difference.

A practical difference in two numbers must be defined and analyzed on a case specific basis. Only in terms of the problem being analyzed does an empirical judgement on the importance of the level of difference make any sense. In some cases a one percent difference may be catastrophic, in others, a fifty percent difference may be acceptable.

For the current research, it is the author's subjective judgement that the metamodel errors of inconsistency are acceptable from the practical perspective. The maximum error reported in the pragmatic plant level validation #3a is 5.35%. Within the bounds of "approximate behavior" this seems reasonable.

APPENDIX E
EMPIRICAL GROUPED CUMULATIVE
DISTRIBUTION FUNCTIONS

EMPIRICAL GROUPED CUMULATIVE DISTRIBUTION FUNCTIONS

In some cases it is preferable to use observed data to specify a sampling distribution for a simulation rather than attempting to fit a theoretical distribution to the data. This type of sampling distribution is known as an empirical distribution. For a continuous random variable, two types of empirical distributions can be defined, one for grouped data and the other for non-grouped data. This research effort utilizes the grouped data approach.

Suppose that there are n observed values of the random variable X . Further, the n X_i 's are grouped into k adjacent intervals $[a_0, a_1)$, $[a_1, a_2)$, ..., $[a_{k-1}, a_k)$, so that the j^{th} interval contains n_j observations, where $n_1 + n_2 + \dots + n_k = n$. For such data, Law and Kelton [1991, sec. 6.2.4] define a reasonable piecewise-linear empirical grouped distribution function G by first letting $G(a_0) = 0$ and $G(a_j) = (n_1 + n_2 + \dots + n_j)/n$ for $j = 1, 2, \dots, k$. Then, interpolating linearly between the a_j 's, the continuous function G becomes:

$$G(x) = \begin{cases} 0 & \text{for } x < a_0 \\ G(a_{j-1}) + \frac{x - a_{j-1}}{a_j - a_{j-1}} [G(a_j) - G(a_{j-1})] & \text{for } a_{j-1} \leq x < a_j \text{ and } j = 1; 2; \dots; k \\ 1 & \text{for } a_k \leq x \end{cases}$$

Generating random variates from the empirical grouped distribution function G can be accomplished using the following inverse transformation algorithm [Law and Kelton 1991, sec. 8.3.12].

- 1) Generate a uniform random variate $U \sim U(0,1)$.
- 2) Find the nonnegative integer J ($0 \leq J \leq k-1$) such that $G(a_J) \leq U < G(a_{J+1})$.
- 3) Return $X = a_J + [U - G(a_J)] (a_{J+1} - a_J) / [G(a_{J+1}) - G(a_J)]$.

Note that J found in step 2 satisfied $G(a_j) < G(a_{j+1})$, so that no X can be returned for an interval for which $n_j = 0$. Also, no X can be returned which does not satisfy $a_0 \leq X \leq a_k$.

APPENDIX F
METAMODEL FILE SPECIFICATIONS

A metamodel file is an ASCII file containing all the necessary information required by the OSU OOM advanced modeling environment to build and subsequently utilize the metamodel. The file is stored under a name whose format is "META.wkc" where "wkc" is a three character acronym for the workcenter name. The two sections below provide (1) the annotated format of the file and (2) the actual metamodel file listing for workcenter OB4.

Annotated MetaModel File Format

number of cells in the grouped empirical CDF (n)
--

cell boundary 1 (time units)

cell boundary 2

.

.

.

cell boundary n+1

rho 1 (percent)

rho 1, cell 1 CDF value (decimal fraction)
--

rho 1, cell 2 CDF value

.

.

.

rho 1, cell n CDF value

rho 2

rho 2, cell 1 CDF value

rho 2, cell 2 CDF value

.

.

.

rho 2, cell n CDF value

.

.

.

File META.OB4 - MetaModel File For WorkCenter OB4

140	24.5	50.0	0.824200	1.000000	1.000000	0.999467
	25.0	50.5	0.929000	1.000000	1.000000	0.999867
0.0	25.5	51.0	0.963467	1.000000	1.000000	0.999933
0.5	26.0	51.5	0.982467	1.000000	1.000000	1.000000
1.0	26.5	52.0	0.990800	1.000000	1.000000	1.000000
1.5	27.0	52.5	0.995733	1.000000	1.000000	1.000000
2.0	27.5	53.0	0.997600	1.000000	1.000000	1.000000
2.5	28.0	53.5	0.999000	1.000000	1.000000	1.000000
3.0	28.5	54.0	0.999533	1.000000	1.000000	1.000000
3.5	29.0	54.5	0.999800	1.000000	1.000000	1.000000
4.0	29.5	55.0	1.000000	1.000000	1.000000	1.000000
4.5	30.0	55.5	1.000000	1.000000	1.000000	1.000000
5.0	30.5	56.0	1.000000	1.000000	1.000000	1.000000
5.5	31.0	56.5	1.000000	1.000000	1.000000	1.000000
6.0	31.5	57.0	1.000000	1.000000	1.000000	1.000000
6.5	32.0	57.5	1.000000	1.000000	1.000000	1.000000
7.0	32.5	58.0	1.000000	1.000000	1.000000	1.000000
7.5	33.0	58.5	1.000000	1.000000	1.000000	1.000000
8.0	33.5	59.0	1.000000	1.000000	1.000000	1.000000
8.5	34.0	59.5	1.000000	1.000000	1.000000	1.000000
9.0	34.5	60.0	1.000000	1.000000	1.000000	1.000000
9.5	35.0	60.5	1.000000	1.000000	1.000000	1.000000
10.0	35.5	61.0	1.000000	1.000000	1.000000	1.000000
10.5	36.0	61.5	1.000000	1.000000	1.000000	1.000000
11.0	36.5	62.0	1.000000	1.000000	1.000000	1.000000
11.5	37.0	62.5	1.000000	1.000000	1.000000	1.000000
12.0	37.5	63.0	1.000000	1.000000	1.000000	1.000000
12.5	38.0	63.5	1.000000	1.000000	1.000000	1.000000
13.0	38.5	64.0	1.000000	1.000000	1.000000	1.000000
13.5	39.0	64.5	1.000000	1.000000	1.000000	1.000000
14.0	39.5	65.0	1.000000	1.000000	1.000000	1.000000
14.5	40.0	65.5	1.000000	1.000000	40	1.000000
15.0	40.5	66.0	1.000000	1.000000	0.000000	1.000000
15.5	41.0	66.5	1.000000	1.000000	0.000000	1.000000
16.0	41.5	67.0	1.000000	1.000000	0.000000	1.000000
16.5	42.0	67.5	1.000000	1.000000	0.000000	1.000000
17.0	42.5	68.0	1.000000	1.000000	0.000000	1.000000
17.5	43.0	68.5	1.000000	1.000000	0.000000	1.000000
18.0	43.5	69.0	1.000000	1.000000	0.000000	1.000000
18.5	44.0	69.5	1.000000	1.000000	0.305800	1.000000
19.0	44.5	70.0	1.000000	1.000000	0.702133	1.000000
19.5	45.0		1.000000	1.000000	0.845067	1.000000
20.0	45.5	25	1.000000	1.000000	0.906000	1.000000
20.5	46.0	0.000000	1.000000	1.000000	0.939133	1.000000
21.0	46.5	0.000000	1.000000	1.000000	0.960800	1.000000
21.5	47.0	0.000000	1.000000	1.000000	0.976800	1.000000
22.0	47.5	0.000000	1.000000	1.000000	0.985267	1.000000
22.5	48.0	0.000000	1.000000	1.000000	0.991867	1.000000
23.0	48.5	0.000000	1.000000	1.000000	0.995267	1.000000
23.5	49.0	0.000000	1.000000	1.000000	0.997600	1.000000
24.0	49.5	0.372267	1.000000	1.000000	0.998933	1.000000

File META.OB4 - MetaModel File For WorkCenter OB4 (continued)

0.000000	1.000000	1.000000	0.550867	1.000000	1.000000	0.855867
0.000000	1.000000	1.000000	0.603667	1.000000	1.000000	0.869600
0.000000	1.000000	1.000000	0.653067	1.000000	1.000000	0.882800
0.000000	1.000000	1.000000	0.699933	1.000000	1.000000	0.895400
0.000000	1.000000	1.000000	0.740467	1.000000	1.000000	0.906867
0.000000	1.000000	1.000000	0.777533	1.000000	1.000000	0.916000
0.120133	1.000000	1.000000	0.811600	1.000000	1.000000	0.924333
0.318400	1.000000	1.000000	0.836200	1.000000	1.000000	0.931200
0.442733	1.000000	1.000000	0.857533	1.000000	1.000000	0.938400
0.531000	1.000000	1.000000	0.877000	1.000000	1.000000	0.945133
0.600733	1.000000	1.000000	0.890867	1.000000	1.000000	0.950333
0.663800	1.000000	1.000000	0.903467	1.000000	1.000000	0.954133
0.718600	1.000000	1.000000	0.915000	1.000000	1.000000	0.957867
0.763667	1.000000	1.000000	0.925267	1.000000	1.000000	0.961133
0.803133	1.000000	1.000000	0.933733	1.000000	1.000000	0.964333
0.836200	1.000000	1.000000	0.941133	1.000000	1.000000	0.967467
0.863867	1.000000	1.000000	0.948000	1.000000	1.000000	0.969933
0.887467	1.000000	1.000000	0.954600	1.000000	1.000000	0.972000
0.905800	1.000000	1.000000	0.959200	1.000000	1.000000	0.974267
0.923933	1.000000	1.000000	0.965067	1.000000	1.000000	0.976467
0.937533	1.000000	1.000000	0.970200	1.000000	1.000000	0.978733
0.949333	1.000000	1.000000	0.974733	1.000000	1.000000	0.980933
0.958333	1.000000	1.000000	0.979467	1.000000	1.000000	0.982733
0.965667	1.000000	1.000000	0.982733	1.000000	1.000000	0.984733
0.972200	1.000000	1.000000	0.985333	1.000000	1.000000	0.986333
0.977467	1.000000	1.000000	0.987600	1.000000	1.000000	0.988200
0.981467	1.000000	1.000000	0.990133	1.000000		0.989667
0.985533	1.000000	1.000000	0.991400	1.000000	80	0.990867
0.988000	1.000000	1.000000	0.992667	1.000000	0.000000	0.991867
0.989733	1.000000	1.000000	0.994200	1.000000	0.000000	0.992533
0.991533	1.000000	1.000000	0.995467	1.000000	0.000000	0.993200
0.992800	1.000000	1.000000	0.995733	1.000000	0.000000	0.994333
0.993200	1.000000	1.000000	0.996400	1.000000	0.000000	0.995133
0.994333	1.000000	1.000000	0.996667	1.000000	0.000000	0.995667
0.994600	1.000000	1.000000	0.997067	1.000000	0.000000	0.996133
0.995133	1.000000	1.000000	0.997600	1.000000	0.076533	0.996600
0.995800	1.000000	1.000000	0.998067	1.000000	0.200467	0.997267
0.996533	1.000000		0.998267	1.000000	0.289867	0.997667
0.997267	1.000000	77.5	0.998467	1.000000	0.359933	0.998000
0.997867	1.000000	0.000000	0.998800	1.000000	0.423800	0.998467
0.998533	1.000000	0.000000	0.999400	1.000000	0.479000	0.998667
0.998867	1.000000	0.000000	0.999667	1.000000	0.532933	0.998867
0.999200	1.000000	0.000000	0.999867	1.000000	0.583467	0.999133
0.999467	1.000000	0.000000	0.999933	1.000000	0.629867	0.999333
0.999667	1.000000	0.000000	1.000000	1.000000	0.668933	0.999667
0.999800	1.000000	0.000000	1.000000	1.000000	0.707533	0.999800
0.999933	1.000000	0.093333	1.000000	1.000000	0.739467	0.999933
1.000000	1.000000	0.241000	1.000000	1.000000	0.767000	0.999933
1.000000	1.000000	0.344267	1.000000	1.000000	0.793400	0.999933
1.000000	1.000000	0.421067	1.000000	1.000000	0.816600	0.999933
1.000000	1.000000	0.488133	1.000000	1.000000	0.837467	1.000000

APPENDIX G
SAS PROGRAM FOR WORKCENTER LEVEL
VALIDATION

WORKCENTER LEVEL VALIDATION - SAS PROGRAM

```

/*
Dissertation Workcenter Level Validation Graphs and Go/No-Go Test
Based on 0.01 Kolmogorov-Smirnov Goodness of Fit Limits
*/

***** Set general parameters;
options ps=58 nodate nonumber;
GOPTIONS DEVICE=PS2EGA;
goptions nodisplay gouttype=independent ftext=simplex htext=1.25;
%let rho=75;
%let trho=0.75;
%let rho2=75;
%let plant=OB1;
libname dbp "c:\dbp\&plant.";
axis1 origin=(10 pct) length=80 pct;

***** Prepare title slide;
proc gslide gout=dbp.wclbm v ;
  footnote1 h=1.00
    "LL and UL are 0.01 Kolmogorov-Smirnov Goodness of Fit Limits";
  footnote2 h=1 ' ';
  footnote3 h=2.00 "WORKCENTER &plant. - Rho &trho.";
run;
footnote1 ""; footnote3 "";

**** Prepare BASE model data;
data one;
  keep x d ;
  set dbp.&plant.d&rho.;
  d=obs_y;
run;
data two;
  keep x j ;
  set dbp.&plant.j&rho.;
  j=obs_y;
run;
data three;
  keep x b ;
  set dbp.&plant.b&rho.;
  b=obs_y;
run;
data four;
  keep x k ;
  set dbp.&plant.k&rho.;

```

WORKCENTER LEVEL VALIDATION - SAS PROGRAM (continued)

```
k=obs_y;
run;
data five;
  keep x s ;
  set dbp.&plant.s&rho.;
  s=obs_y;
run;
data avg&rho.;
  keep x c_avg avg model ul ll;
  retain c_avg 0;
  merge one two three four five ;
  by x;
  cellwidth=0.5;
  avg = (d+j+b+k+s)/5;
  c_avg = c_avg + (avg*cellwidth);
  if c_avg > 1 then c_avg=1;
  temp=c_avg;
  model='BASE';
  output;
  d_01 = 1.63/sqrt(3000);
  ul=c_avg+d_01;
  ll=c_avg-d_01;
  if ul>1 then ul=.;
  if ll<0 then ll=.;
  c_avg=ul; model='UL ';output;
  c_avg=ll; model='LL ';output;
  c_avg=temp;
run;

***** Prepare META model data;
data one;
  keep x d ;
  set dbp.&plant.md&rho2.;
  d=obs_y;
run;
data two;
  keep x j ;
  set dbp.&plant.mj&rho2.;
  j=obs_y;
run;
data three;
  keep x b ;
  set dbp.&plant.mb&rho2.;
  b=obs_y;
```


WORKCENTER LEVEL VALIDATION - SAS PROGRAM (continued)

```

run;
data four;
  keep x k ;
  set dbp.&plant.mk&rho2.;
  k=obs_y;
run;
data five;
  keep x s ;
  set dbp.&plant.ms&rho2.;
  s=obs_y;
run;
data avg&rho.m;
  keep x c_avg avg model;
  retain c_avg 0;
  merge one two three four five ;
  by x;
  cellwidth=0.5;
  avg = (d+j+b+k+s)/5;
  c_avg = c_avg + (avg*cellwidth);
  if c_avg > 1 then c_avg=1;
  model='META';
run;

***** Plot CDF graph;
DATA ALLcrvs1;
  set avg&rho. avg&rho.m;
  if x > 20 then delete;
run;
PROC GPLOT DATA=ALLCRVS1 gout=dbp.wclbm;
  TITLE1 H=2.0 "Cummulative Distribution Function";
  LABEL X='TIME-IN-SYSTEM' C_AVG='CDF';
  SYMBOL1 V=NONE I=JOIN l=33 ;
  symbol2 v=NONE I=JOIN l=3 ;
  symbol3 v=NONE I=JOIN l=1 ;
  symbol4 v=NONE I=JOIN l=3 ;
  PLOT c_avg*x=model /
    haxis=axis1
    name="&rho. CDF";
RUN;

***** Plot PDF graph;
DATA ALLcrvs2;
  set allcrvs1;
  if model='UL ' then delete;

```

WORKCENTER LEVEL VALIDATION - SAS PROGRAM (continued)

```

if model='LL ' then delete;
run;
PROC GPLOT DATA=ALLCRVS2 gout=dbp.wclbmrv;
  TITLE1 h=2.0 "Probability Density Function";
  LABEL X='TIME-IN-SYSTEM' AVG='PDF';
  SYMBOL1 V=NONE I=JOIN l=33 ;
  symbol2 v=NONE I=JOIN l=1 ;
  symbol3 v=NONE I=JOIN l=4 ;
  symbol4 v=NONE I=JOIN l=5 ;
  PLOT avg*x=model /
    haxis=axis1
    name="&rho. PDF";
RUN;

***** Quantitatively Test for Limit Violations;
data base (KEEP=BASE X) meta (KEEP=META X) UL (keep=ul x) LL (keep=ll x);
  set allcrvs1;
  if model='BASE' THEN DO;
    BASE = C_AVG;
    OUTPUT BASE;
  END;
  if model='META' THEN DO;
    META = C_AVG;
    OUTPUT META;
  END;
  if model='UL ' THEN DO;
    OUTPUT UL;
  END;
  if model='LL ' THEN DO;
    OUTPUT LL;
  END;
RUN;
DATA TEST;
  MERGE BASE META UL LL ;
  BY X;
RUN;
data _null_;
  set test end=eof;
  if ll=. then ll=0;
  if ul=. then ul=1;
  if meta < ll then do;
    put 'Lower K-S Limit Violated at ' x= meta= ll=;
    stop;
  end;
end;

```

WORKCENTER LEVEL VALIDATION - SAS PROGRAM (continued)

```
if meta > ul then do;  
  put 'Upper K-S Limit Violated at ' x= meta= ul=;  
  stop;  
end;  
if eof=1 then put 'No K-S Limit Violations Detected';  
run;
```

VITA

David B. Pratt

Candidate for the Degree of

Doctor of Philosophy

Thesis: DEVELOPMENT OF A METHODOLOGY FOR HYBRID META-MODELING OF HIERARCHICAL MANUFACTURING SYSTEMS WITHIN A SIMULATION FRAMEWORK

Major Field: Industrial Engineering and Management

Biographical:

Personal Date: Born in Oklahoma City, Oklahoma, January 6, 1954, the son of Harold F. and G. Lahoma Pratt. Married Jan M. Hussey on August 28, 1976. Father of two children, Brian David born September 21, 1979 and Kristi Marie born October 12, 1981.

Education: Graduated from John Marshall High School, Oklahoma City, Oklahoma in May 1972; received Bachelor of Science Degree in Industrial Engineering and Management from Oklahoma State University in May, 1976; received Master of Engineering Degree in Industrial Engineering from Oklahoma State University in July, 1977; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in May, 1992.

Professional Experience: Operations Research Analyst, Phillips Petroleum Company, May, 1976 to March, 1981; Senior Operations Research Analyst, Phillips Petroleum Company, April, 1981 to September, 1985; Section Director, Operations Research, Phillips Petroleum Company, October, 1985 to May, 1986; Senior Production Systems Engineer, Garrett Turbine Engine Company, July, 1986 to January, 1988; Manager, Operations Research, International Paper Company, February, 1988 to November, 1988; Research Associate, School of Industrial Engineering and Management Oklahoma State University, November, 1988 to December, 1992; Assistant Professor, School of Industrial Engineering and Management Oklahoma State University, January, 1992 to present.