1

DISCRETE EVENT SIMULATION

MODEL DECOMPOSITION

THESIS

Scott R. Matthes
Captain, USAF

AFIT/GOR/ENS/88D-13

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GOR/ENS/88D-13

DISCRETE EVENT SIMULATION MODEL DECOMPOSITION

THESIS

Scott R. Matthes
Captain, USAF

AFIT/GOR/ENS/88D-13

AFIT/GOR/ENS/88D-13

DISCRETE EVENT SIMULATION MODEL DECOMPOSITION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment fo the

Requirements for the Degree of

Master of Science

Scott R. Matthes, B.S.

Captain, USAF

December 1988

Acknowledgements

This thesis is a result of the efforts and encouragements of many people. I would like to thank all who contributed. Special thanks are due to my thesis advisor, Major Kenneth Bauer, for his significant contributions and insights into this research. My reader, Dr. James Chrissis, also merits thanks for his efforts in making this thesis a readable document.

I also would like to thank my wife, Denise, for helping me to put things in proper perspective. Her gentle nudges to work harder were needed when I was approaching the burn-out stage. I also want to thank my father, mother, and little sister for their support during these past 19 months.

Most of all, I want to express my gratitude to the Lord Jesus Christ for His mercy and faithfulness. It is indeed true that "the faithfulness of the Lord endures forever" (Psalm 117:2b). Praise the Lord!

Scott R. Matthes

Table of Contents

List of Figures

List of Tables

AFIT/GOR/ENS/88D-13

Cont'd from 1473

Abstract

This research focused on interpreting a discrete event simulation model's condition specification primitives and their associated actions. A network representation was created using these condition action pairs (CAPs) as network nodes. The arcs, or edges, of the network represent information being transferred such as specific attributes of the CAPs. This network representation was decomposed into smaller networks, or sub-networks, by taking advantage of the structure of the network. The structure of the network was translated via a software interface into an edge-incidence matrix (E-matrix). The E-matrix was then transformed into a pseudo-covariance matrix (C-matrix). The C-matrix was used in the creation of a SAS data set which served as the input necessary to do principal components analysis. Two examples were used to demonstrate this procedure. Theses. (AW)

DISCRETE EVENT SIMULATION MODEL DECOMPOSITION

I. Introduction

Background

Simulation models are currently being used for a multitude of purposes. Some simulation models are concise and well organized thereby facilitating their usage. However, some of these models may be quite lengthy and complex which causes development costs to rise beyond an acceptable level. The creation of an effective development environment has helped to eliminate this problem. Tools which are used to support modeling and analysis have been incorporated into this environment. Such an environment uses a model specification language which has condition specifications as its basic constructs, Condition specifications and their associa~~ted actions~~ tions are called condition action pairs (CAPs).

Cont'd pg. viii

One distinct advantage of using these condition specifications is that it allows the developer of a discrete event simulation to establish an intermediate form of the simulation while the simulation is being developed. This intermediate form is a bridge between the conceptual stage and the executable stage. When progressing from the concept of the simulation to its intermediate form, objectives of the simulation are identified. Each objective necessitates the completion of one or more processes. The particular steps of these processes need to then be established. At this time, the condition specifications of these steps are constructed, and the complete simulation is formulated as a single model. This intermediate form is useful in determining if there are errors in the design of the simulation. Finding errors early in the development of a simulation is both cost effective and time efficient for the developer.

Another use of this intermediate form is in establishing the structure of the model. The condition specifications can be transformed into a network representation. A network representation can then be developed directly from the definitions of the systems and objectives of the simulation. This network representation allows for the building of a run-time-efficient implementation as well as the design of statistical analysis procedures. A network representation can be obtained by expressing the CAPs as network nodes. This network representation will then be decomposed into smaller networks, or sub-networks, to take advantage of the structure of the network. The arcs, or edges, of the sub-networks represent information that is being transferred, such as specific attributes of the CAPs. The structure of each sub-network is translated into an edge-incidence matrix (E-matrix), and this E-matrix is converted into a pseudo-covariance matrix (C-matrix). The C-matrix can then be used in the creation of an IMSL or SAS data set. Computer aids are not generally available for this type of procedure. However, once this procedure is completed the data set will then be used as the input necessary to analyze the network and its structure.

Currently, a network representation of a simulation can only be accomplished by a non-automated effort. For even relatively small networks, this manual effort is extremely labor intensive. Upon completion of the development of the network representation, the network can be decomposed. The decomposition of the network is also a laborious task. Because of the degree of manual effort involved in this entire process, errors are likely to occur. These errors obviously make the results inaccurate and can, in fact, render the entire effort useless. If the errors are not discovered, as is often the case, potentially misleading work on the simulation continues.

Once the simulation has been properly decomposed, analysis can be performed to identify the sub-networks. This analysis usually requires mathematical and/or statistical software. To use these software packages, information obtained from the structure of the sub-networks must be manually transferred to a file and formatted to be compatible with the software package that provides the computations.

Problem Specification

Representing a discrete event simulation model's condition specifications and their associated actions as a network is an error-prone, labor-intensive process. Analysis of the network representation necessitates examining the structure of the network and its sub-networks. This analysis is often inaccurate due to the size, complexity, and/or sophistication of the network.

Scope of the Problem

This research focuses on interpreting a discrete event simulation model's condition specifications and their associated actions. This procedure was translated into a computer program to facilitate its use. One statistical procedure, principal components analysis, was used for the analysis phase.

A simple existing simulation model was used for gathering preliminary information. A computer program was then developed to automate the procedure of interpreting the CAPs. A more complex model was then used as a test case upon completion of the computer program. This was done to verify the program's ability to create a network representation of a simulation model and determine its structure.

Research Objectives

The primary objective of this research was to represent a discrete event simulation model as a network and then decompose that network into sub-networks. The structure of the network representation was translated into an edge-incidence matrix. This entire process was accomplished by means of a computer program. This program will greatly aid the analysis of the discrete event simulation model.

A secondary objective of this research was to develop some basic error checking techniques to ensure the accuracy of the condition specification and its network representation. Diagnostics generated by the computer program provide information necessary to correct any deficiencies that are discovered, such as storing information that is not used anywhere in the model.

Overview

This research reviewed literature concerning the concepts of model decomposition and computer specification language. A model decomposition methodology was then developed and translated into a computer program. Two examples were then decomposed and analyzed to demonstrate the effectiveness of the methodology and the computer program. Potential areas of related research were also identified in this research.

II. Literature Review and Background

Introduction

The concept of model decomposition is but one major area that provided insight into potential methods for pursuing this research. Other areas of interest include computer specification language and computer assisted modeling. Network theory was also applied throughout the solution process. Information on network structures and methods of their decomposition prove to be beneficial in this research.

Related Literature and Background

Formal logic has been used in the field of mathematics for problem solving to enhance the creation of prototype models. Defined relationships between entities allows for the use of a specification language in model definition. The specification language facilitates the rapid development of prototypes by combining object programming and logic programming (de Freitas and others, 1986:844-849).

Two early attempts to automate the formulation of a discrete event simulation model led to the development of an expert system. The first attempt led to a prototype system that used formal logic but was not flexible. This inflexibility led to the second prototype system which used a more informal approach (Doukidis and Paul, 1985:79-90). The experience obtained from these efforts helped to alleviate potential problems. While some constraints are to be expected when using a computer program, such as an expert system, minimizing the degree of awkwardness and the inconvenience to the user is a major consideration. To create the computer program to interpret condition specifications, some syntax requirements will be necessary. However, these will be as flexible as possible.

The idea of using a model development environment (MDE) was introduced to try to alleviate some of the modeling costs. This was done by bridging the gap between the model as it was conceived and the actual executable representation of the model. As a result, condition specifications are well defined in this approach (Overstreet and Nance, 1985:190-201). This set of condition specifications was one of the cornerstones for this research. Pre-defined conventions were observed to the greatest extent possible.

A model development environment is used for rapid prototyping in the requirements specification phase. The operating system and hardware are major parts of this environment and can be thought of as a first layer of the MDE. A kernel MDE is a second layer to be followed by a minimal MDE and then the MDE itself (Balci, 1986:53-67).

The Conical Methodology can be used in evaluating software development methods. This methodology entails analyzing the objectives, principles, and attributes of a discrete event simulation model. The manner in which these elements are linked together is evaluated as a key component of these software development methods. The method is compared with the software tools available that comprise the environment (Nance, 1987:38-43). Despite the fact that this was initially developed for large-scale simulation model development, the concepts are applicable in part to smaller models.

A method developed to represent modular discrete event models in distributed simulator architectures was extended to facilitate modular, hierarchical model specification. These specifications are transformed into a logical form called abstract simulator architecture (Zeiglar, 1985:3-7). This architecture can be evaluated to shed light on the decomposition of the network representation of the CAPs into modules or sub-networks.

Once the CAPs have been decomposed into their primitives and properly interpreted, the network representation is complete. A framework was then designed to facilitate statistical analysis. Principal components analysis, a special case of factor analysis, allows for finding the linear combination of factors that explains the greatest part of the variance in the data. This was the method of statistical analysis used. Usually the magnitude of the eigenvalues obtained from a correlation matrix determine how many factors are to be considered significant.

Kaiser developed a criterion which is the most frequently used method for determining the number of factors to retain. This criterion states that components whose eigenvalues are greater than unity should be retained. However, this research uses a covariance matrix instead of a correlation matrix because all of the units in the system are identical, allowing more information to be obtained from the results. The numerical value of the information is not converted into a range from -1 to +1. Instead, the values of the covariance matrix are used. These values represent actual information in the network representation. Also, the values in the covariance matrix have dimensions associated with them. This enhances the interpretability of the results. Because the covariance matrix was analyzed, Kaiser's criterion was not used. Instead, the number of factors to be retained was based on a specified percentage of the variance of the data explained. Component loadings were used to determine which factors were significant and what attributes comprise these factors (Dillon and Goldstein, 1984:23-107).

This method is demonstrated using a simple directed graph as in Figure 1. Arcs represent information that is being transmitted from one node to another. These arcs are used in the formulation of an

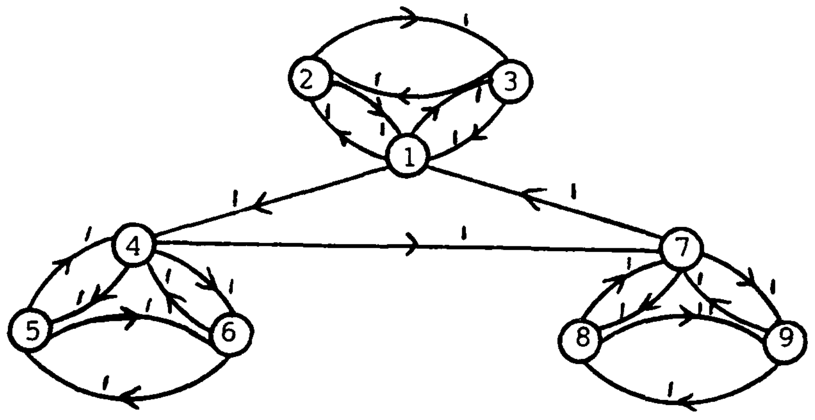

Figure 1 - Simple Directed Graph

edge-incidence matrix (E-matrix). The E-matrix can be converted into a pseudo-covariance matrix (C-matrix) where $C = E \cdot E^T$. Different C-matrices may be obtained by applying various weighting schemes to the problem. This might be done to account for the number of arcs going from one node to another or to account for the amount of information being passed along the arcs. Weighting schemes are translated into a weight matrix (W-matrix). When this is done, $C = E \cdot W \cdot E^T$. If no specified weighting scheme is used, the W-matrix defaults to the identity matrix. Once a C-matrix is obtained, factor analysis can then be used to determine the principal components of the system. In this example, a pseudo-correlation matrix was used. Therefore, the number of factors to be retained was determined by using Kaiser's criterion. Three factors are retained. Using the varimax rotation during factor analysis yields the factor loadings shown in Table 1. Three nodes load on each of the factors, as circled in the table. Notice that the nodes that load on each factor interact heavily with each other. These factors defines the structure of the network. These interactions illustrate how this directed graph can be decomposed into three distinct sub-graphs. This information provides a low complexity solution to the model decomposition problem (Bauer and others, 1985:185-188).

Table 1. Rotated Factor Loadings

Node	Factor 1	Factor 2	Factor 3
1	.08294	.08294	.75994
2	-.02441	-.02441	.80475
3	-.02441	-.02441	.80475
4	.75994	.08294	.08294
5	.80475	-.02441	-.02441
6	.80475	-.02441	-.02441
7	.08294	.75994	.08294
8	-.02441	.80475	-.02441
9	-.02441	.80475	-.02441

The CAPs are composed of condition specifications and their associated actions. The condition specification is an attempt to relate a general description to a formal set of guidelines. These condition specifications are either Boolean expressions or time-based signals. The condition specification primitives are the basic constructs which make up the formal set of guidelines. The primitives can be treated as statements in a computer language. Each primitive has a specific function with which it is associated. An example of a CAP is the following statement:

When Alarm(end_repair)

In this case, "When Alarm" is the name of the primitive, and the function "end_repair" is a time-sequencing condition. Initialization and termination primitives must be present when formulating a series of CAPs to be interpreted. Each primitive must be written in a specific manner using precise punctuation. This syntax facilitates the interpretation of the primitive (Nance and Overstreet, 1986:1-14). Nance and Overstreet illustrate this concept with an example. This example was used as a test case for the computer program that was developed in this research.

One means of measuring the complexity of a simulation model is by using the control and transformation metric. This metric uses a network representation and is based on the number of arcs and variables that communicate between nodes. Model complexity (MC) is defined as:

$$MC = \sum_{i=1}^{n} (2 * RW_i + 1.5 * W_i + R_i) * A_i / N$$

where

RW_i	= number of variables in node i read and written
W_i	= number of variables in node i writen only
R_i	= number of variables in node i read only
A_i	= number of arcs entering or leaving node i
N	= total number of nodes

This metric may be of interest in the network decomposition process (Wallace, 1987:597-602).

An example of using condition specifications as the bridge between the conceptual model and the executable model is derived from the illustration of Computerized Manufacturing Systems (Lenz and Talavage, undated:3-8). This illustration provides for a moderately in-depth application of condition specifications. Doubly-subscripted variables are the norm, and triply-subscripted variables are not uncommon in this derived set of condition specifications. This led to an opportunity to use different weighting schemes based on the amount of information, contained within the subscripts, communicated between nodes when obtaining the C-matrix (Talavage, 1986:1-3). This example of condition specifications was used as a second test case due to its level of complexity.

Properly diagnosing condition specifications is important. Because this diagnosis is not automated, errors may be overlooked. Thus, it is important to automate diagnostic functions to ensure that the condition specifications are properly interpreted. Three primary categories in which automated diagnostic support is needed are analytical functions, comparative functions, and informative functions. Analytical functions aid in determining the existence of a property of a model representation. Comparative functions measure the differences among multiple model representations. Informative functions yield characteristics that can be extracted or derived from model representations (Nance and Overstreet, 1987:590-595). The research herein provides automated support in the diagnostic areas of analytical and informative functions. The analytical functions addressed are attribute utilization, attribute consistency, and connectivity. The informative functions addressed are attribute classification and decomposition.

Using a computer specification language facilitates model decomposition. Methods of model decomposition currently exist that were used in this research. The decomposed network can then be analyzed using available statistical techniques. A methodology that combined these aspects was developed and formed into a computer program.

III. Methodology

Introduction

A methodology was developed to decompose a discrete event simulation model. The first step was to interpret the model's CAPs. After this interpretation, a network representation was developed. This network was then analyzed using principal components analysis. This process was then automated by developing a computer program to perform the steps of this methodology.

Decomposition Procedure

A condition specification for a model consists of interface specifications, object specifications, report specifications, and transition specifications. The interface specifications identify the input and output attributes. Attributes record information about the objects of the simulation that are needed for the model. They also have values assigned to them to account for changes in the state of the associated objects. Object specifications consist of the object name, a list of attributes associated with the particular object, and a range for each of the attributes. Report specifications are produced to provide information about the behavior of the model. Because the complexity of the data collection and computation process varies, the form of this specification is not prescribed. Transition specifications are made up of ordered pairs called condition action pairs (CAPs). The transition specification may also include a set of functions which are used to simplify expressions in both the conditions and actions. The construct of ordered pairs is a property which facilitates the decomposition of a discrete event simulation into a network representation. Each CAP has a condition and an associated action. If the condition of the CAP is true, then the associated action occurs. The action continues to occur until the condition is no longer true. Several CAPs may have conditions which are true at the same time. CAPs that have identical conditions are grouped into an action cluster (Overstreet and Nance, 1985:190-201).

Interpreting CAPs is the foundation for decomposing a discrete event simulation into a network representation. This process begins by determining the type of attribute used in a transition specification statement.

The three types of attributes that exist in the transition specification are control, input, and output attributes. An attribute is a control attribute of an action cluster if it appears in the condition expression of the action cluster. The first statement in an action cluster is a CAP. The condition of the CAP is either a "WHEN" or "WHEN ALARM" statement. By identifying the statement as the beginning of a new action cluster, the attribute in the statement is determined to be a control attribute. The input attribute is an attribute that affects one or more output attributes of the associated action cluster. Output attributes of an action cluster are those attributes which can be changed by the actions within the action cluster.

Consider the following simple action cluster:

```
WHEN ALARM(need_re-stock);
quantity :== quantity + resupply;
END WHEN;
```

The attribute "need_re-stock" is a control attribute. When this condition exists, this action cluster will become active. The first "quantity" in the second statement is an output attribute. Its value is incremented (changed) by the action that occurs within this action cluster. The second statement has two input attributes -- "quantity" and "resupply." Both of these input attributes affect the value of the attribute "quantity" on the left side of the assignment statement (the second statement in this action cluster).

In addition to an assignment statement, several other types of statements may exist in a transition specification. The statements which can be identified by the computer program developed in this research are summarized in Table 2.

Table 2. Types of Statements

Statement	Syntax
Assignment	output :== input expression
Set Alarm	SET ALARM(alarm name[[argument list]], time delay)
Create	CREATE(object type[, object id])
Destroy	DESTROY(object type[, object id])
When Alarm	WHEN ALARM(alarm name expression[, parameter list])
New Node	{name of a new action cluster to follow}
Comment	any statement that does not follow the above conventions

Each action cluster can be thought of as a node in a network representation. Each attribute within an action cluster is identified as being part of the composition of a node. After each action cluster has had all of its attributes identified, communication between action clusters is determined. Communication occurs when the value of an output attribute of a node is used in another node's control or input attributes. These communications are represented as arcs in a network representation. The value of the attribute that is being communicated represents the information that is flowing along the arcs of the network.

The network representation can be represented in an edge-incidence matrix (E-matrix). The dimension of the E-matrix is the number of nodes by the number of arcs or edges. Each column of this matrix consists of two ones and the rest zeroes. The ones are placed in the rows corresponding to the nodes which the arc connects. If two or more columns are identical, then all but one of them are discarded. This yields an E-matrix in which all of the columns are unique. An example of an E-matrix in which nodes one and two, one and three, and two and three communicate is:

$$\begin{matrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix}$$

where each column represents an edge, and the rows represent the different nodes.

Once the E-matrix has been constructed, the network representation (NR) can be derived by the equation $NR = E \cdot E^T$. It is often desirable to apply a weighting scheme to the network representation equation to account for various features of the network such as the number of arcs between two nodes or the amount of information being passed between two nodes. The dimension of the weighting matrix (W-matrix) is the number of edges by the number of edges. The W-matrix is a diagonal matrix. Thus, the generalized network representation equation becomes $NR = E \cdot W \cdot E^T$. If no weighting scheme is desired, an identity matrix can be assumed as the W-matrix. This allows the generalized network representation to be used.

The resulting NR-matrix is symmetric and positive semi-definite. Because of this, it can be treated as if it is a pseudo-covariance matrix (C-matrix). Eigenvalues and their associated eigenvectors can be extracted from this C-matrix. Using the eigenvalues, an examination is made to determine if it is feasible to reduce the dimensionality of the C-matrix. This examination is performed by using principal components analysis. The Statistical Analysis System (SAS) software was used to do the principal components analysis. The results of the principal factors analysis were interpreted in an attempt to shed light on the inherent structure of the network representation.

Automating the Procedure

This decomposition process was automated by developing a computer program. This program was written in the language BASIC. The major reasons for choosing this language were the language's ability to manipulate strings of data, the researcher's familiarity with the language, and portability. This program was developed on an IBM/AT-compatible personal computer running under MS-Dos 3.20.

This program was developed in three stages that directly parallel the decomposition process. These stages were 1) identify action clusters and their associated attributes, 2) create a network representation from the information obtained in the first stage, and 3) create the appropriate C-matrix to analyze using principal components analysis.

The transition specification, which is in a standard ASCII text file, is read by the program to identify the action clusters and their associated attributes. Each line of the transition specification is categorized into one of the statement types given in Table 2. The type of statement is identified by examining the characters in the statement. The entire statement is read into one variable. After this is done, a search is made for strings of characters that uniquely identify the statement type. This is accomplished by searching for key words that appear in each statement type. For example, the string "WHEN ALARM" only appears in a "WHEN ALARM" statement. In this manner the statement types are identified. For an assignment statement, the string ":==" is the object of the search. The beginning of a node is determined by finding a "{" character. Once the "{" has been found, the expression inside the braces is assumed to be the name of that particular node. If the line of the transition specification being examined is not one of the statements found in Table 2, then the statement is considered to be a comment statement.

If the statement is identified to be from Table 2, the attributes of the statement are then identified. This is also done by examining the characters in the statement. If the statement is a "WHEN" or "WHEN ALARM" statement, the control attribute(s) is identified according to the syntax of the statement. Input attributes may be present in these two statements if the condition is given as an expression. If the statement is an assignment statement, the output attribute is to the left of the assignment indicator (":=="). Input attributes are to the right of the assignment indicator. It is quite possible to have more than one input attribute in the same statement, particularly when the input side of the statement is given as a mathematical expression. The "CREATE" statement has only one attribute and that is an output attribute. Likewise, the "DESTROY" statement has one attribute; however, this is an input attribute. The "SET ALARM" statement contains both output and input attributes. The alarm name is an output attribute, and the time delay is an input attribute.

When an attribute is found, it is entered into an array of attributes. This is a three dimensional array consisting of the node in which the attribute is found, the type of attribute that is found, and a number representing the specific occurrence of that type of attribute in that particular node. This is done so that it is possible to determine which attributes are passed from one node to another.

After all of the attributes of a transition specification have been obtained, the network representation of the transition specification is created. To create the network representation, the arcs between the nodes need to be established. This is accomplished by taking an attribute from one node and checking all of the other nodes for a matching attribute. If a match is detected, the type of attribute is checked to ensure that communication is viable. Viable communication occurs when an output attribute matches a control or input attribute of another node. However, matching attributes do not communicate with each other if they are the same type (ie, both output) of attribute. Input and control attributes do not communicate with each other because neither attribute causes a change in the other.

Once all of the arcs have been established, the E-matrix is created. The E-matrix (nodes by edges) is initialized as a zero matrix. A specific column of the E-matrix is associated with each arc as the arc is identified. Rows are identified with the nodes in the network. Thus, E(2,5) represents the value of node two and edge five. A value of one is placed in the element if it corresponds to one of the nodes that is being connected by the arc for the given edge. If the same arc is identified more than once, only one column is placed into the E-matrix. Duplicate arcs are temporarily discarded. The network representation is obtained by matrix multiplication; $NR = E \cdot E^T$.

The C-matrix is obtained by using the generalized network representation formula, $NR = E \cdot W \cdot E^T$. This is also a simple matrix multiplication operation. The W-matrix (edges by edges) is initialized as a zero matrix. The element of the W-matrix that provides a weight for a particular edge is the element that is on the diagonal of the W-matrix and is in the appropriate column. For example, if the third edge needs to have a weight of two, then the element W(3,3) = 2. Each weight is determined and then placed in the diagonal of the W-matrix.

This research allows for the application of three different weighting schemes. The first case is the simplest; the identity matrix is used as the W-matrix (W1 = I). The resulting NR-matrix (nodes by nodes) yields a general description of the network. The elements on the diagonal of the matrix indicate the number of nodes with which a particular node communicates. For example, if element NR(3,3) = 5, then node three communicates with five other nodes in the network representation. Off-diagonal elements indicate whether or not communication exists between the node represented by the column and by the row. In this case, the element is either a one or a zero, a one indicates that communication exists. The sum of the elements of a row or column, excluding the element on the diagonal, is equal to the element that is on the diagonal in that particular row or column.

The second weighting scheme (W2) examined in this research takes into account the number of attributes that are communicated between two nodes. While the E-matrix consists of only one entry for a specific edge, the diagonal of the W-matrix is the number of occurrences of that particular edge. Suppose that nodes one and three form the fifth edge in the E-matrix. If three different attributes are communicated from node one to node three, then the element W(5,5) = 3. The NR-matrix that results from using this weighting scheme is similar to the NR-matrix that resulted in the first case. The elements on the diagonal of the NR-matrix indicate the number of arcs that communicate to and from the particular node associated with the element. For example, if element NR(4,4) = 7, then seven arcs exist between node four and the other nodes. Unlike the previous case, off-diagonal elements may have any non-negative value. If, for example, element NR(4,2) = 2, then two arcs exist between node two and node four. This indicates that two different attributes are being passed between these nodes. As in the previous case, the sum of the elements of a row or column, excluding the element on the diagonal, is equal to the element that is on the diagonal in that particular row or column.

The third weighting scheme (W3) examined in this research takes into account the actual number of pieces of information being passed throughout the network representation. Subscripts are often used to indicate how many of a particular attribute exist. In a transition specification, these subscripts are often generalized by using a variable as the subscript. Subscripts are identified as part of the attribute. Usually, the communicating attributes have the same subscripts. On occasion, however, they do not match exactly. When this happens, the subscript of the control or input attribute is used instead of the subscript that is associated with the output attribute. This is because only the quantity of information used in the control or input attribute is needed regardless of how much information of a particular kind is available. Suppose that aircraft[i] is an output attribute and that aircraft[1] is a control attribute. Only the value of aircraft[1] would be passed along the edge connecting the nodes of these attributes. If all of the subscripts in a transition specification were equal to one, then this weighting scheme would be the same as in the second case. The W-matrix is formed as in the previous case except that the number of pieces of information transferred along each arc is taken into account instead of merely counting the arcs. If three arcs exist between nodes two and six, then the quantity of each arc's information is added together and placed on the appropriate diagonal. For example, suppose aircraft[i], pilot[j], and loadmaster[k] are the attributes being passed from node two to six. Now suppose that in this particular case i = 3, j = 4, and k = 5. Suppose further that the eighth column in the W-matrix represents the edge existing between nodes two and six; the value of element W(8,8) = 12. The NR-matrix that results from using this weighting scheme is similar to the NR-matrix that resulted in the second case. The diagonal elements of the NR-matrix indicate the number of pieces of information that are communicated to and from the particular node associated with the element. For example, if element NR(7,7) = 17, then 17 pieces of information are being communicated between node seven and the other nodes. Off-diagonal elements may have any non-negative value. If element NR(7,5) = 3, then three pieces of information are being passed between nodes seven and five. Again, the sum of the elements of a row or column, excluding the element on the diagonal, is equal to the element that is on the diagonal in that particular row or column.

Once the NR-matrix is obtained, it is used as the basis for the principal components analysis. The NR-matrix is actually the C-matrix used in this analysis. For this analysis, SAS will be used. This necessitates that the C-matrix be placed in a file that can be used by SAS. Two options exist for this. The first option is to merely create an ASCII data file that contains the C-matrix. In this case, the file that causes SAS to perform the principal components analysis must read in the data file. The second option is to create a SAS file that contains the C-matrix without having to call it in from another file. This research utilized option two because it is simpler to transfer one file than two. The need for this will be discussed shortly.

The SAS software has a procedure for doing factor analysis. In this procedure is an option that enables an analyst to do principal components analysis. Several principal component options are included in the SAS file that are needed for analyzing the obtained C-matrix. The first option used is to specify that a covariance matrix is to be analyzed rather than a correlation matrix. This covariance matrix is the C-matrix. The correlation matrix derived from the C-matrix could be used, but because all of the entries in the C-matrix are based on the same units, the covariance matrix is used. Another option used is the "number of factors to be retained" feature. This is used in lieu of Kaiser's criterion because a covariance matrix is being used. The computer program that creates the SAS file asks the user to input this number. The last option used in the SAS file is a varimax rotation. This option yields a set of rotated component loadings. This is useful because it prevents all of the component loadings from loading on the first few factors. This rotation maximizes the variance explained by each of the factors. These component loadings are then examined for interpretation. This aids in the explanation of the network being considered.

An example of the SAS file that is created by the computer program is given in Figure 2. Any lines that begin with an asterisk are comment lines. Comments are used to remind the user which type of weighting scheme was employed for this particular SAS run. They are also used when the weighting scheme accounts for each piece of information as the comment statements identify the value given to each of the subscripts.

Diagnostics

Two ASCII files in addition to the "NODE.CMP" file will be found in the currently logged directory of the user's computer when the program has finished executing. These files, "INF.DAT" and "ERRORS.DAT", aid the user in interpreting the results from the computer program.

```
OPTIONS LINESIZE = 80;
*;
* E x E';
*;
DATA CONDSPEC;
INPUT N1 N2 N3 N4;
CARDS;
      2  1  1  0
      1  2  0  1
      1  0  1  0
      0  1  0  1
;
PROC PRINT;
PROC FACTOR    COV    NFACTORS = 2   ROTATE = VARIMAX ;
     VAR  N1  N2  N3  N4;
```

Figure 2. Sample SAS File

The file "INF.DAT" contains information about the network representation of the transition specification. Each attribute is listed and the edge that it forms is given by the numbers of the associated action clusters (node numbers) for reference. The type of attribute is also given in this file. A numbering system has been established for convenience. Control attributes are identified by the number one. Input attributes are identified by the number two, and output attributes are identified by the number three. The number of the attribute type is listed next to the node number of the associated attribute. Also given in this file is the number used in the W-matrix that corresponds to the specified edge. If the same edge is encountered more than once, the number corresponding to the last time an edge is found will be used in the W-matrix. Notice that two numbers for the W-matrix are given per attribute. The first of these is the value used in the second weighting scheme, and the second is the value used in the third weighting scheme. No number is needed for the first weighting scheme because the identity matrix is used. An example of what an entry in this file looks like is

```
aircraft    ,     3   1            7   3             4      27
```

In this case, the attribute "aircraft" is found in nodes three and seven. In node three, it is a control attribute. In node seven, it is an output attribute. The number four represents the fourth occurrence of the edge connecting nodes three and seven. The number 27 represents the quantity of information being passed along this particular edge. At the end of the "INF.DAT" file is listed the number of unique edges that exist in the network representation.

The file "ERRORS.DAT" contains error messages that have been generated throughout the course of the computer program's execution. This program identifies two types of inconsistencies so that the user will have the information necessary to make sure that he has a valid transition specification. In this research, a valid transition specification is defined to be a transition specification in which there exist no unmatched attributes. Unmatched attributes are attributes that do not form an edge anywhere in the network representation. Attributes that are key words or reserve words are exempt from this restriction.

The first type of inconsistency that is recorded in the error file deals with subscripts. Subscript errors occur in two distinct ways. The first way in which a subscript error can happen is if an attribute that creates an edge does not have the same dimensions in both of the connecting nodes. For example, attribute "automobile[i,j]" is incompatible with attribute "automobile[i]" because the former is two dimensional and the latter is one dimensional. Should this occur, it must be concluded that one of the attributes is incorrect, and an error message is written to the error file. This message lists the attribute with the subscripts and the two nodes involved in the edge. The message also lets the user know that this is a fatal error because it results in an incorrect assessment of the transition specification. Even though this is a fatal error, the program will continue to run. This allows the user to have a complete error file so that he may make all of necessary changes at one time.

The second way in which a subscript error can occur is when the subscript does not agree in the connecting nodes. For example, attribute "part[i]" does not have exactly the same subscript as attribute "part[j]" does. This may or may not be by design. Because this may be what was intended, a warning message will be written to the error file indicating that the subscripts are not an exact match. However, the user must determine the accuracy of the statements in the transition specification. One case where this may be what is desired is when a numerical value is given in a subscript. Attribute "part[i]" may actually communicate with "part[3]" in the transition specification. This edge will not be considered in the case where both subscripts are different numbers. For example "part[2]" does not communicate with "part[3]" in this environment.

The second type of inconsistency recorded in the error file exists when an attribute in a node does not communicate with another node anywhere in the transition specification. This may occur because the user changed the name of the attribute in the middle of his work and failed to change all occurrences of that attribute. Perhaps a more likely reason for this error is that a typographical error was committed when entering the transition specification. When this inconsistency is discovered, the name of the attribute is written to the error file. The node where this attribute is located is given for reference. The type of attribute (control, input, or output) is also given. This is not considered a fatal error because this inconsistency may or may not be intentional. It could result from neglecting to include an attribute in with the key words or reserve words that should have been put into one of these categories. The computer program continues to create the desired network representation until completed.

Syntax Requirements

It was the intent of this research to keep the syntax requirements of the transition specification to a minimum. Another goal was to keep them in harmony with previously published requirements. Any changes made were for the reason of creating a standardized requirement or simplifying an existing requirement. One major requirement is that each statement end with a semicolon. This convention is used because it greatly simplifies locating the end of a statement. The basic syntax requirements are listed in Table 2. However, a more in depth explanation of them is given in this section.

Subscripts are placed in square brackets, [], for ease of interpretation. This is so that they can be readily distinguished from called routines such as "poisson(time)" and statements such as "WHEN ALARM(load_cargo)" which use parentheses. If an attribute has more than one dimension, no space is used between the subscripts. For example, "part[i,j]" is acceptable whereas "part[i, j]" is not.

The assignment statement is relatively free formatted. The assignment indicator is ":==" for this statement. Usually a space precedes and follows the assignment indicator. On the left of the indicator is the attribute to which a value is being assigned. The value is on the right of the indicator. This value may be a numerical value or an expression involving attributes. The expression may use -, +, *, or / as operators. If the expression will not fit on a single line, a second line may be used with no continuation character necessary.

The "SET ALARM" statement is more complex than the assignment statement. This statement is distinguished by the "SET ALARM" which occurs at the beginning of the statement. The parameters that are associated with the statement are included in parentheses. There is no space between the "M" and the "(" in this statement. The first parameter in the statement is the name of the alarm to be set. This is an output attribute and may, of course, contain subscripts. The second parameter is the length of time required to set this alarm. This may be a numerical value or an attribute. If it is an attribute, it is of the input variety. A ", " separates these two parameters. Notice that a space exists after the comma. Both of these parameters are required in this statement.

The "CREATE" and "DESTROY" statements are nearly identical to each other. The "CREATE" statement is distinguished by the word "CREATE" which occurs at the beginning of the statement. The word "DESTROY" indicates that particular statement. Two parameters are included inside parentheses. Again, no space is allowed between the opening parenthesis and the last letter of the word preceding it. The first parameter is the object or attribute that is being either created or destroyed. This attribute is given without the subscript. The second parameter is the subscript that is associated with the attribute. Again, a comma followed by a space separates the two parameters. However, in this case the second parameter is not required. If it is not used, simply place the closing parenthesis after the first parameter. Do not place a comma after the first parameter in this case. An output attribute is associated with the "CREATE" statement, and an input attribute corresponds to the "DESTROY" statement.

The "WHEN ALARM" statement is distinguished by the words "WHEN ALARM" at the beginning of the statement. Its parameters are enclosed in parentheses with the same restrictions as the other statements. The first parameter is an expression involving an attribute. This expression may be given in two forms. The first form is merely naming the attribute. For example, "WHEN ALARM(eject[i]);" enables this action cluster whenever "eject[i]" is encountered. The second form of the expression involves relational operators. For example, "WHEN ALARM(eject[i] = 2);" enables this action cluster when the value of "eject[i]" = 2. A space precedes and follows the relational operator. Two conditions may be in the same statement if connected by an ampersand. For example, "WHEN ALARM(eject[i] = 2 & fire[i] = true);" is a valid statement. A space precedes and follows the ampersand. The attribute on the left of the operator is a control attribute while the one of the right is an input attribute.

A statement that is related to the "WHEN ALARM" statement is the "WHEN" statement. However, this statement does not use parentheses. It does have a conditional expression that activates the statements in the action cluster. This expression follows the same syntax as in the "WHEN ALARM" statement. It also may have two conditions in the same statement.

A new action cluster or node is distinguished by an opening brace, {, at the beginning of the statement. What follows the brace is taken to be the name of the node. A closing brace, }, designates the end of the node name. If the node is the initialization or termination point in the transition specification, it should be so named. When the node names "Initialization" or "initialization" or "Termination" or "termination" are found, the action cluster is disregarded in the decomposition process. This is because often these nodes communicate with a majority of the nodes merely to start or finish the simulation. Therefore, these nodes lend no structure to the model. This phase of the syntax structure is slightly different than what has been done in the past. However, it does not deviate from the former conventions to an extent that this researcher considered unacceptable.

Comment statements are the last type of statement to be considered. This is a free formatted statement with one exception. It cannot conform to the syntax of one of the previously discussed statements. These statements are read by the computer program but not considered in the decomposition process.

The methodology developed in this research was tested using two different transition specifications. The results from the testing, given in the next chapter, verify the procedures that were used in this technique. A simple guide on how to use the software developed can be found in Appendix A. The actual BASIC code is listed in Appendix L.

IV. Analysis of Example Transition Specifications

Introduction

Two example transition specifications were decomposed in this study. The first example was a simple transition specification which, when represented as a network, consisted of six nodes. The model in the second example had 15 nodes in its network representation. In each of these examples, the three different weighting schemes were used yielding three distinct C-matrices for each example. The C-matrices were then independently analyzed using principal components analysis.

The simple example was also used to examine the effects of considering the time delay involved in the flow of information from one node to another. In this brief look at how time affects the network structure, the only time delay taken into account was the delay incurred from the setting of an alarm until the alarm was activated.

A Simple Example

The first transition specification examined was the Repairman Example (Nance and Overstreet, 1986:11). The transition specification was slightly modified so that it conformed to the syntax required to utilize the newly developed decomposition software. This transformed transition specification can be found in Appendix B.

The program provided the diagnostics given in Appendix C. An illustration of the network representation for this transition specification is given in Figure 3. The action clusters or nodes for this transition specification were defined as shown in Table 3 below. When prompted for a subscript value, the value five was used. This is the number of facilities in the simulation. Each of the weighting schemes was used (option seven of the matrix multiplication menu). The SAS file that was created by the software is given in Appendix D. This file was uploaded to a Vax 11/785 using Xmodem protocol. The SAS file was executed and yielded the results given in Appendix E.

Table 3. Repairman Example Nodes

Node	Node Name
1	Failure
2	Begin repair
3	End Repair
4	Travel to idle
5	Arrive idle
6	Travel to facility

For the first of the three matrix multiplications (E · E'), four factors explained 92.21 percent of the variance of the model. Three factors explained only 75.24 percent of the variance, so four factors were retained for the analysis. After the varimax rotation, the nodes that loaded on each factor are shown in Table 4.

Table 4. Rotated Factor Pairings (Example 1 - Case 1)

Factor	Nodes
1	End repair, Arrive idle
2	Failure, Begin repair
3	Travel to facility
4	Travel to idle

The six nodes have been decomposed into four factors which when interpreted provide insight into the structure of the network representation. The "Arrive idle" node occurs when the repairman has completed all of the repair work that has been called in, arrives at his

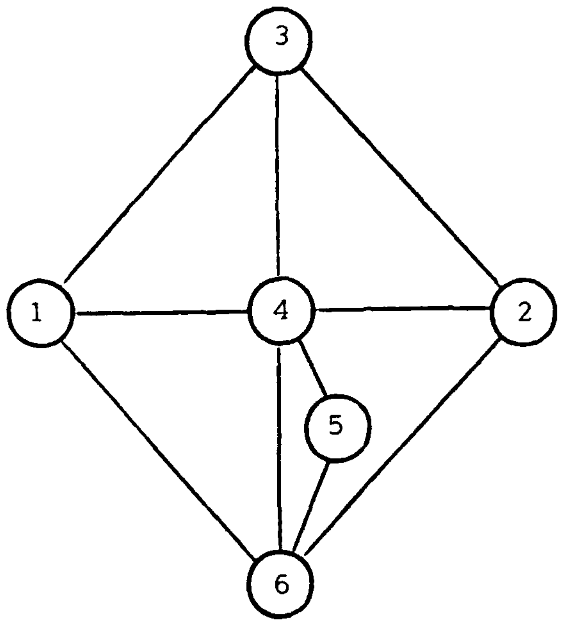

Figure 3. Network Representation of Repairman Example

office, and is waiting to go on another call. The fact that this node is paired with "End repair" makes perfect sense. One possible name for factor one is "Finished work orders" because the two nodes combined represent just that. The second factor combines the "Failure" and "Begin repair" nodes. This is logical because the beginning of a repair is quite dependent upon the time that the system fails. "Work needed" is a possible name for the second factor. This is because work is required when a system fails and when the repair process is started. Because factors three and four have only one node associated with them, a simple way to name these factors is just to retain the node name as their factor name.

The number of factors chosen was based on retaining the fewest factors which would sufficiently explain the structure of the network. This explanation included a logical interpretation of the factors as well as explaining a reasonable amount of the model's variance. More than one SAS run was made to acquire this information. When nodes loaded consistently on factors in more than one SAS run, a degree of structure was indicated in the model.

For the second of the three matrix multiplications (E · W2 · E'), three factors explained 83.98 percent of the variance of the model. Two factors explained only 69.70 percent of the variance, so three factors were retained for the analysis. After the varimax rotation, the nodes that loaded on each factor are given in Table 5.

Notice that in this case, the "Failure" node does not load on any factor. This node had weak loadings on the first and second factors, but never had a loading with a magnitude greater than or equal to 0.5

which was used as the criterion for factor loading. As in the previous case, factor one is comprised of the "End repair" and "Arrive idle" nodes. This factor can again be labeled as the "Finished work orders" factor. The second factor in this case combines the "Begin repair" and "Travel to facility" nodes. This is reasonable because the beginning of a repair depends on when the repairman travels to the facility where the repair work is needed. "Work needed" is again a possible name for the second factor. This is because work is still outstanding while a repairman is traveling to the repair site and when the repair process is started. Because factor three has only one node associated with it, the obvious choice for a factor name is the same as the node name.

Table 5. Rotated Factor Pairings (Example 1 - Case 2)

Factor	Nodes
1	End repair, Arrive idle
2	Begin repair, Travel to facility
3	Travel to idle

For the third of the three matrix multiplications ($E \cdot W3 \cdot E'$), three factors explained 86.63 percent of the variance of the model. Two factors explained only 71.65 percent of the variance, so three factors were retained for the analysis. After the varimax rotation, the nodes that loaded on each factor are given in Table 6.

As in each of the previous cases, factor one is comprised of the "End repair" and "Arrive idle" nodes. This factor can again be label as the "Finished work orders" factor. The second factor in this case combines the "Travel to idle" and "Travel to facility" nodes. This is logical in light of the fact that a repairman will often have to go to his office to await a call. Once a call is received, he then must travel to the facility to do his job. "Between work orders" is a possible name for the second factor. The third factor "Failure" and "Begin repair" has an obvious relationship. These two nodes combine into a factor which could be called the "Time awaiting repairman" factor.

Table 6. Rotated Factor Pairings (Example 1 - Case 3)

Factor	Nodes
1	End repair, Arrive idle
2	Travel to idle, Travel to facility
3	Failure, Begin repair

The three different weighting schemes used in this simple example provide similar, although not exactly the same, results. The factors may be compared in Table 7. The factor names could very well have been something else as they are merely an interpretation of the rotated factor pairings. These factors break the network representation of the transition specification into sub-networks which could be used as a means of distributing work load in the development of the full simulation model.

Table 7. Repairman Example Factor Names

Factor	Case 1	Case 2	Case 3
1	Finished work orders	Finished work orders	Finished work orders
2	Work needed	Work needed	Between work orders
3	Travel to facility	Travel to idle	Time awaiting repairman
4	Travel to idle		

A More Complicated Example

The second transition specification examined was the computerized manufacturing systems model (Talavage, 1986:1-3) or Manufacturing Example. The transition specification was modified so that it conformed to the syntax required to utilize the newly developed decomposition software. This transformed transition specification can be found in Appendix F.

The program provided the diagnostics given in Appendix G. Despite the fact that several fatal errors occurred, the analysis was continued because this research was not concerned with the correctness of the simulation, but rather with illustrating how a simulation can be decomposed. An illustration of the network representation for this transition specification is given in Figure 4. The action clusters or nodes for this transition specification were defined as shown in Table 8. When prompted for a subscript value, the value five was used. Each of the weighting schemes was used (option seven of the matrix multiplication menu). The SAS file that was created by the software is given in Appendix H. This file was uploaded to a Vax 11/785 using Xmodem protocol. The SAS file was executed and yielded the results given in Appendix I.

For the first of the three matrix multiplications ($E \cdot E'$), six factors which were retained for analysis explained 81.96 percent of the variance of the model. After the varimax rotation, the nodes that loaded on each factor are given in Table 9.

Table 8. Manufacturing Example Nodes

Node	Node Name	Alias
1	operation completed for part i on machine j	opcomm
2	part i to off-shuttle position k of machine j	offshu1
3	part i in off-shuttle position k of machine j	offshu2
4	operations completed for part i	opcom
5	next operation for part i	nextop
6	next station for next operation for part i	nextsta
7	find cart for part i	findcart
8	move cart j to pickup part i	mvcart
9	cart j arrives to pickup part i	cartarrp
10	part i arrives onto cart j	part-cart
11	cart j arrives to drop off part i at machine k	cartarrd
12	part i arrives into on-shuttle	onshu
13	part i can move into machine	mchld1
14	part i moves into machine	mchld2
15	next part from queue selected to move into machine	mchld3

The 15 nodes have been decomposed into six factors which, when interpreted, provide insight into the structure of the network representation. The four nodes that make up factor one are related to each other as they are the completion of a part's cycle in the model. One possible name for factor one is "Part Operations Completed" because the four nodes combined represent just that. The second factor is composed of nodes that deal with the getting a cart ready so that it can carry a part from one location to another. At this point, a cart is

Table 9. Rotated Factor Pairings (Example 2 - Case 1)

Factor	Nodes
1	opcomm, offshu2, opcom, nextop
2	findcart, mvcart, part-cart
3	nextsta, cartarrp, cartarrd
4	mchld1, mchld2
5	offshu1
6	onshu

found, summoned, and then loaded. The name "Get Cart for Part" can be applied to this factor. The third factor involves moving the part from one machine to another. A cart picks up the part and moves it to the next station where it drops it off. The name "Move Part to Next Station" can be applied to factor three. The fourth factor combines the "mchld1" and "mchld2" nodes. This makes sense because both of these activities are required to load a part into a machine. "Load Part Into Machine" is a possible name for the fourth factor. Because factors five and six have only one node associated with them, the obvious method of naming these factors was to give them their respective node name. The node "mchld3" did not load well on any of the six factors retained in this analysis.

For the second of the three matrix multiplications (E · W2 · E'), six factors retained for analysis explained 85.84 percent of the variance of the model. After the varimax rotation, the nodes that loaded on each factor are given in Table 10.

Table 10. Rotated Factor Pairings (Example 2 - Case 2)

Factor	Nodes
1	nextop, mvcart, cartarrp, part-cart
2	cartarrd, onshu
3	opcom, mchld2, mchld3
4	opcomm, nextsta, findcart
5	offshu1, offshu2
6	mchld1

The four nodes that make up factor one are related to each other as each is a step in a process to get a part ready for the upcoming operation. One possible name for factor one is "Prepare Part for Next

Operation" because the four nodes that are combined perform this function. The second factor is composed of two nodes which together enable a part to arrive at the on-shuttle. The name "Part Arrives On-Shuttle" can be applied to this factor. The third factor encompasses the completion of operations for a part. The name "Part Operations Completed" can be applied to factor three. The fourth factor consists of the "opcomm", "nextsta" and "findcart" nodes. These three nodes ensure that a part is ready for the next operation. "Part Ready for Next Operation" is a name that could be given to this factor. This factor is slightly different than factor one. Factor one is a process of preparing a part for the next operation, whereas factor four is indicative of a part that is ready for the next operation. The fifth factor combines the "offshul" and "offshu2" nodes. This make sense because both of these activities are required for a part to be in the off-shuttle position of a machine. "Part Off-Shuttle" is a possible name for the fifth factor. Factor six has only one node associated with it, so again the factor was given the same name as its only node.

For the last of the three matrix multiplications (E · W3 · E'), seven factors retained for analysis explained 99.61 percent of the variance of the model. After the varimax rotation, the nodes that loaded on each factor are given in Table 11.

In this case, the 15 nodes were decomposed into seven factors. The two nodes that combine to become factor one make a logical pair because dropping a part off at a machine and the part going into the on-shuttle position are activities which are closely related. One possible name for factor one is "Part Arrives On-Shuttle." The second factor is

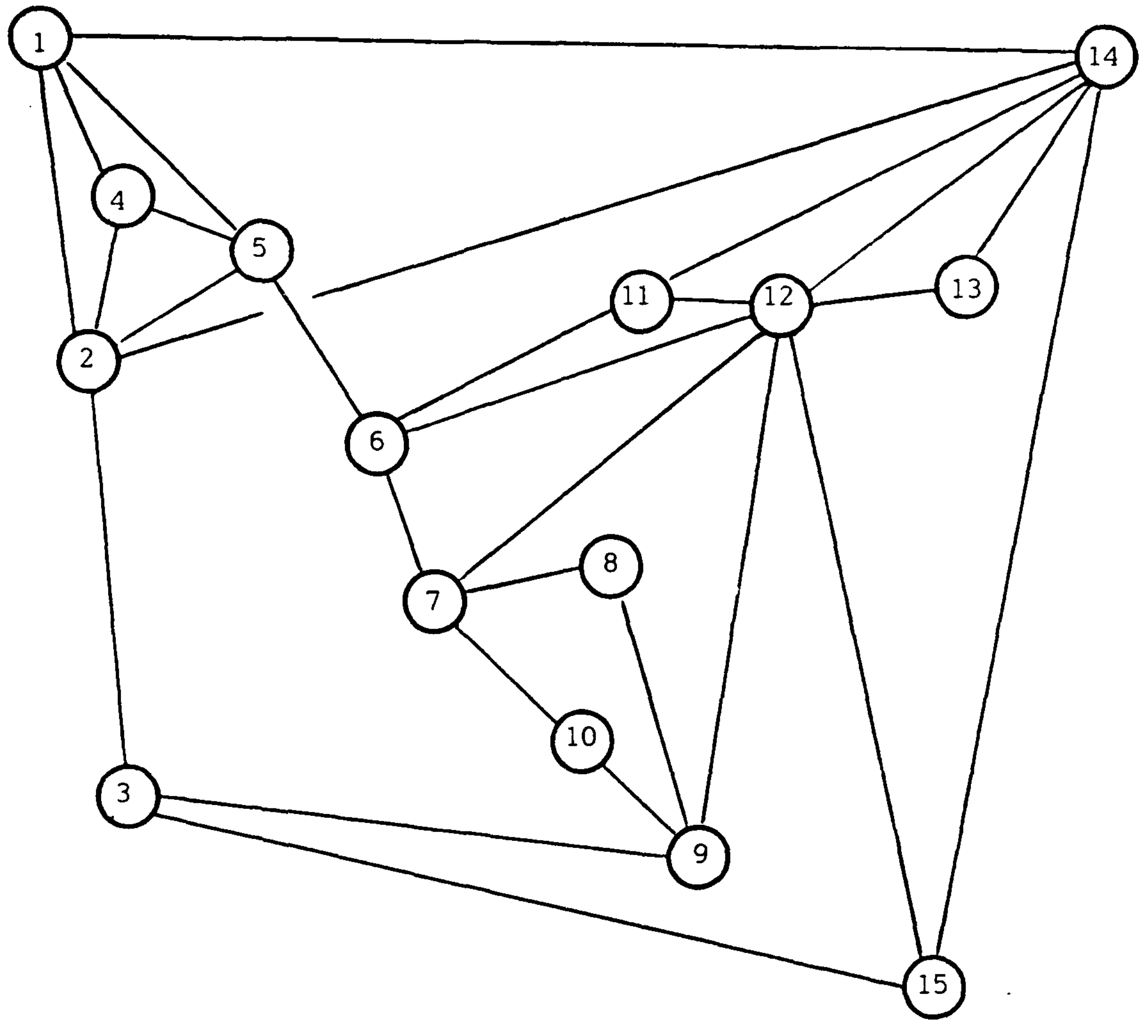

Figure 4. Network Representation of Manufacturing Example

composed of two nodes that deal with the getting a cart ready for its next operation. The nodes "nextop" and "nextsta" combine to form a factor which could be called "Part Ready for Next Operation." The third factor includes four factors that prepare a part to be placed in the off-shuttle position of a machine. These factors include loading the part into a cart and putting the part in its position. The name "Part Off-Shuttle" can be applied to factor three. The fourth factor is one in which the cart is obtained for the part to be transported. "Get Cart" is one name for factor four, which is made up of the "findcart" and "mvcart" nodes. The fifth factor combines the "mchld2" and "mchld3" nodes. This makes sense because both of these activities are required to load a part into a machine. "Load Part Into Machine" is a possible name for the fifth factor. Once again, two of the factors have only one node associated with them. By previous convention, these factors (six and seven) keep the node name as their factor names. The node "opcom" did not load well on any of the seven factors retained in this analysis.

Table 11. Rotated Factor Pairings (Example 2 - Case 3)

Factor	Nodes
1	cartarrd, onshu
2	nextop, nextsta
3	cartarrp, part-cart, offshul, offshu2
4	findcart, mvcart
5	mchld2, mchld3
6	opcomm
7	mchld1

The three different weighting schemes used in this example provide similar, although not exactly the same, results. Table 12 summarizes the factor interpretations. These factor names could very well have been something else as they are merely an interpretation of the rotated factor pairings. As in the previous example, these factors break the network representation of the transition specification into sub-networks which could be used as a means of distributing work load in the development of the full simulation model. Instead of having to work directly with 15 nodes, the model developer need only work with the number of factors involved in a specified case.

Table 12. Manufacturing Example Factor Names

Factor	Case 1	Case 2
1	Part Operations Completed	Prepare Part for Next Operation
2	Get Cart for Part	Part Arrives On-Shuttle
3	Move Part to Next Station	Part Operations Completed
4	Load Part Into Machine	Part Ready for Next Operation
5	Part to Off-Shuttle Position	Part Off-Shuttle
6	Part Arrives On-Shuttle	Part Can Move Into Machine

Factor	Case 3
1	Part Arrives On-Shuttle
2	Part Ready for Next Operation
3	Part Off-Shuttle
4	Get Cart
5	Load Part Into Machine
6	Part Operation Completed on Machine
7	Part Can Move Into Machine

A Brief Examination of Time-Based Signals

Another type of weighting scheme is one based on the length of time information takes to travel along the edge between nodes. In the "SET ALARM" statement the second parameter is the time required until the alarm is activated. This parameter was used as an input (t) to a weighting function. This function was Weight = 10 $\cdot$ $e^{-.025t}$. This function was chosen because more emphasis was desired on events that happen in the near future versus the distant future. The value 10 is a scaling constant, and the value -.025 reflects the rate of declining emphasis placed on time-based information.

The Repairman Example examined previously was again used. It has four "SET ALARM" statements. Each of these required an input for the time delay. A mean of 520 hours was chosen as the time delay between the repairing of a system and its failure. This number was based on 13 weeks at 40 hours per week of system utilization. The mean time chosen for travel to the facility and travel back to the office was one hour. The amount of time required to perform the repair work needed was estimated to be three hours. These times were entered into the weight function as needed. A new C-matrix was then obtained to analyze. Because this involves the pieces of information flowing on an edge, the W-matrix used was merely a modified version of the W-matrix used in the third case in the Repairman Example considered previously. The SAS input for this example is in Appendix J.

Three factors explained 83.37 percent of the variance of the model. After the varimax rotation, the nodes that loaded on each factor are given in Table 13.

The six nodes have been decomposed into the same three factors which occurred in the third case of the simple example. (The SAS output from this example is in Appendix K.) This is not totally unexpected due to the small number of nodes and time-based signals in the network. It was anticipated that this weighting scheme would shed some light on which of the three world views would provide the best means of formulation -- event scheduling, activity scanning, or process interaction. Because the time-based signal results do not differ from the results that were not a function of time, the results obtained may indicate that the Repairman Example is not sensitive to time and should be considered from the event scheduling viewpoint.

Table 13. Rotated Factor Pairings (Time-Based Example)

Factor	Nodes
1	End repair, Arrive idle
2	Travel to idle, Travel to facility
3	Failure, Begin repair

V. Findings

Conclusions

Representing a discrete event simulation model's condition specification as a network is beneficial. This network can be decomposed so that the individual developing the model can take advantage of the structure of the model. Once the network representation is translated into an edge-incidence matrix, the developer can perform so matrix calculations in order to obtain a pseudo-covariance matrix (C-matrix). The C-matrix was then used as the input data set during principal components analysis.

Applying various weighting schemes to the C-matrix enabled the analyst to have a better understanding of what exactly was happening in the network representation. Using the first weighting scheme resulted in obtaining information necessary to illustrate the network representation. The second weighting scheme provides the analyst with information concerning the number of attributes that are being sent along each edge in the network. The third weighting scheme yields results which enable a network representation to illustrate the flow of every piece of information in the simulation.

Performing principal components analysis on the C-matrix enabled the analyst to ascertain what the key factors were in the network representation. These factors formed logical clusters of nodes which decomposed the model into meaningful sub-models. Once these factors have been determined, the results are interpreted based on their factor composition.

Automating the decomposition process tremendously enhances the overall effectiveness of analyzing a discrete event simulation model. This eliminates the numerous mistakes that often occur when not taking advantage of the computer's ability to extract data and perform calculations without error.

Several types of diagnostics have been included in the computer program. These diagnostics provide the user with feedback on the composition of the network representation of the model as well as alert the analyst to any inconsistencies that may be present in the transition specification.

Recommendations

This research has opened several potential areas for further research. Improving diagnostics is one area of interest. Being able to correctly create a transition specification is vital to doing any kind of work in the area of model decomposition. It is important to detect errors at the earliest possible time so that the analyst does not continue to work in an errant direction with potentially misleading results. Real-time error handling and correction procedures could be added to computer programs that will further automate the decomposition process. Other possible enhancements may become obvious as the computer program developed in this research is utilized.

Another potential area in which enhancements to this research may be made is the incorporation of additional transition specification statements into the decomposition software. Two types of statements that were not included in this research are the "AFTER ALARM" and "CANCEL ALARM" statements. These and any statements which may be developed in the future can and should be incorporated into a computer program, perhaps as an extension to the one developed in this research.

An additional area of interest is the manner in which W-matrices are constructed. Weighting schemes play an important role in the decomposition process. This is, perhaps, the area with the greatest potential for improvement. One particular weighting scheme that should be examined in the future is weighting edges based on the length of time it takes to pass information along a given edge. An extremely brief attempt at employing this type of scheme was developed in this research. Much more attention should be given to this area, particularly with large examples.

One final area of interest is the comparison between the sub-models that are obtained from a transition specification. Although the difference in the sub-models is slight, the manner in which the sub-models would be partitioned out to computer programmers would vary. When these sub-models are assimilated into the final simulation model, the efficiency and validity of the model should be compared to those aspects of a model that is comprised of similar, but different, sub-models.

Appendix A. How to Implement the Computer Program

Once a SAS file has been created, it needs to be uploaded to a mainframe computer which has SAS resident on it. This step is necessary to do the principal components analysis. Uploading the SAS file is done by using a file transfer program. Two common means by which to upload files from a personal computer to a mainframe are using the Kermit or Xmodem file transfer programs. After the SAS file has been uploaded to the mainframe, make sure that SAS has been invoked on the mainframe. If the SAS filename is NETWORK.SAS, then the file may be processed by entering the statement "SAS NETWORK" into the computer. This is but one way to run SAS. This may vary among mainframe computer systems. It is best to check with people who are familiar with the local computer facility.

Naturally it is necessary to first create the SAS file so that it can be uploaded. To do this requires implementing the computer program written as a part of this research. After placing the floppy disk into the disk drive, type "DECOMP" to begin the program.

The first input that the user is required to give is the number of key words used in the transition specification. Key words are any words which the user does not wish to have classified as an attribute. For example, "system_time" would normally appear as an input attribute. "System_time" is an attribute that will not affect the network representation because it is used only as an input attribute. Therefore it is not possible to make an edge with this attribute. Once the number of key words has been entered, the user is asked to input those key words into the computer. This must be exactly what the user intends. This program does not ignore the case of letters. Upper case and lower case letters are not interpreted to be the same. If the user enters a "0" for the number of key words, he will not, of course, be prompted to enter any key words. An ASCII file called "RESERVE.WRD" has been created for key words that frequently occur in various transition specifications. To add words to this file the user must edit the file before running the "DECOMP" program. Key words may also be deleted from this file as desired.

After the user has finished inputting information about key words, the computer will prompt the user for the name of the transition specification to be analyzed. This transition specification needs to have been created before running this program. It also has to be in ASCII format. The program will then open the file containing the transition specification and read it into memory.

Once this operation is completed, the action clusters and their associated attributes are identified. These are written to an ASCII file called "NODE.CMP" (a diagnostic file) and also displayed on the screen of the personal computer. After the user has finished viewing the last action cluster's attributes, the program examines each of the attributes for any subscript notation. If one or more subscripts are found, a determination is made to find out if the subscript is a variable. If it is a variable, then the user will be asked to input the maximum value of the subscript. For example, if the attribute "aircraft[i]" is found, then the "i" is determined to be a subscript. The program realizes that this is not a numeric value, so it prompts the user for a value. If 13 aircraft were involved in this transition specification, the user would enter a value of 13 at this point. After all of the subscript maximum values have been entered, the program finds the network representation of the transition specification. The E-matrix is then created from the network representation. Once the E-matrix has been created, a menu appears on the screen giving the user a choice of which weighting scheme is to be employed. This menu is illustrated in Figure 5. All three weighting schemes may be examined with only one run of this program. It does not matter in the development of the SAS file if more than one weighting scheme is chosen. Only one SAS file will be created, but the desired principal components analysis will be performed on each of the chosen weighting schemes. The user is prompted by the program to supply a name for the SAS file. After the user chooses which weighting scheme he will use, he is prompted to enter the desired percentage of the variance to be explained. This will determine how many factors are retained in the principal components analysis. The SAS file is then created using the user's input for both the weighting scheme and the number of factors to be retained. As each C-matrix is calculated, the results are printed on the computer screen. After viewing these results, type any key to continue the program. After the program finishes writing the SAS file, the program is completed.

```
                    OPTIONS OF MATRIX MULTIPLICATIONS

1.  E x E'

2.  E x W2 x E'

3.  E x W3 x E'

4.  BOTH 1 & 2

5.  BOTH 1 & 3

6.  BOTH 2 & 3

7.  ALL 3  (1 & 2 & 3)

ENTER CHOICE
```

Figure 5. Matrix Multiplication Menu

Appendix B. Repairman Example Transition Specification

```
{Initialization}
    WHEN Initialization
       INPUT(n, max_repairs, mean_uptime, mean_repairtime );
       CREATE repairman AN object WITH
          location :== idle;
          status:== avail;
          END WITH
       CREATE facility A p_set WITH INDEX i : 1..n ALSO
          failed[i] :==false;
          total_downtime[i] :== 0;
          SET ALARM(failure[i], neg_exp(mean_uptime));
          END WITH;
       DEFINE down A d_set OF facility WITH failed[i] = true;
       num_repairs :== 0;
    *  END WHEN

{Termination}
##  WHEN num_repairs r max repairs
       FOR i :== 1 TO n DO
          percent downtime[i] :== total downtime[i] / system time;
          OUTPUT(i, total downtime[i],);
       *  END FOR
       STOP
    *  END WHEN

{Failure}
    WHEN ALARM(failure[i]);
       failed[i] :== true;
       begin_downtime[i] :== system_time;
    *  END WHEN

{Begin repair}
    WHEN ALARM(arr_facility[i]);
       SET ALARM(end_repair[i], neg_exp(mean_repairtime));
       status :== busy;
       location :== i;
    *  END WHEN

{End repair}
    WHEN ALARM(end_repair[i]);
       SET ALARM(failure[i], neg_exp(mean_uptime));
       failed[i] :== false;
       total_downtime[i] :== total_downtime[i] +
          (system_time - begin_downtime[i]);
       status :== avail;
       num_repairs :== num_repairs + 1;
    *  END WHEN
```

```
{Travel to idle}
##  WHEN failed[i] = true & status = avail & location <> idle;
       SET ALARM(arr_idle, traveltime(location, idle));
       status :== travel;
    *  END WHEN

{Arrive idle}
    WHEN ALARM(arr_idle);
       status :== avail;
       location :== idle;
    *  END WHEN

{Travel to facility}
    WHEN status = avail & failed[i] = false;
       i :== closest_failed_fac(failed[i], location);
       SET ALARM(arr_facility[i], traveltime
          (location, i));
       status :== travel;
    *  END WHEN
```

Appendix C. Repairman Example Diagnostics

NODE.CMP

```
  NODE  3     Failure

CONTROL ATTRIBUTE                failure[i]
 OUTPUT ATTRIBUTE                failed[i]
 OUTPUT ATTRIBUTE                begin_downtime[i]


  NODE  4     Begin repair

CONTROL ATTRIBUTE                arr_facility[i]
 OUTPUT ATTRIBUTE                end_repair[i]
  INPUT ATTRIBUTE                mean_repairtime
 OUTPUT ATTRIBUTE                status
  INPUT ATTRIBUTE                i
 OUTPUT ATTRIBUTE                location


  NODE  5     End repair

CONTROL ATTRIBUTE                end_repair[i]
 OUTPUT ATTRIBUTE                failure[i]
  INPUT ATTRIBUTE                mean_uptime
 OUTPUT ATTRIBUTE                failed[i]
  INPUT ATTRIBUTE                total_downtime[i]
  INPUT ATTRIBUTE                begin_downtime[i]
 OUTPUT ATTRIBUTE                total_downtime[i]
 OUTPUT ATTRIBUTE                status
  INPUT ATTRIBUTE                num_repairs
 OUTPUT ATTRIBUTE                num_repairs


  NODE  6     Travel to idle

CONTROL ATTRIBUTE                failed[i]
CONTROL ATTRIBUTE                status
CONTROL ATTRIBUTE                location
 OUTPUT ATTRIBUTE                arr_idle
  INPUT ATTRIBUTE                location
 OUTPUT ATTRIBUTE                status


  NODE  7     Arrive idle

CONTROL ATTRIBUTE                arr_idle
 OUTPUT ATTRIBUTE                status
 OUTPUT ATTRIBUTE                location
```

```
   NODE  8     Travel to facility

  CONTROL ATTRIBUTE              status
  CONTROL ATTRIBUTE              i]
    INPUT ATTRIBUTE              failed[i]
    INPUT ATTRIBUTE              location
   OUTPUT ATTRIBUTE              i
   OUTPUT ATTRIBUTE              arr_facility[i]
    INPUT ATTRIBUTE              location
    INPUT ATTRIBUTE              i
   OUTPUT ATTRIBUTE              status
```

ERRORS.DAT

```
NON-MATCHING SUBSCRIPTS

NODE          ATTRIBUTE                          NODE          ATTRIBUTE


ATTRIBUTES WITHOUT A COUNTERPART

       ATTRIBUTE                                 NODE      TYPE


mean_repairtime                                    4      INPUT
mean_uptime                                        5      INPUT
total_downtime                                     5      INPUT
num_repairs                                        5      INPUT
total_downtime                                     5      OUTPUT
num_repairs                                        5      OUTPUT
```

INF.DAT

```
     ATTRIBUTE                         NODE   C/I/O     NODE   C/I/O    W1     W2
failure                              ,   3      1         5      3      1       5
failed                               ,   3      3         6      1      1       5
failed                               ,   3      3         8      1      1       5
begin_downtime                       ,   3      3         5      2      2      10
arr_facility                         ,   4      1         8      3      1       5
i                                    ,   4      2         8      3      2       6
end_repair                           ,   4      3         5      1      1       5
status                               ,   4      3         6      1      1       1
status                               ,   4      3         8      1      3       7
location                             ,   4      3         6      1      2       2
location                             ,   4      3         8      2      4       8
failed                               ,   5      3         6      1      1       5
failed                               ,   5      3         8      1      1       5
```

```
status                               ,  5     3        6     1      2      6
status                               ,  5     3        8     1      2      6
status                               ,  6     1        7     3      1      1
status                               ,  6     1        8     3      1      1
location                             ,  6     1        7     3      2      2
arr_idle                             ,  6     3        7     1      3      3
status                               ,  6     3        8     1      2      2
status                               ,  7     3        8     1      1      1
location                             ,  7     3        8     2      2      2

NUMBER OF EDGES = 22
```

Appendix D. Repairman Example SAS Input File

```
OPTIONS LINESIZE=80;
*;
*   E x E';
*;
DATA   CONDSPEC;
INPUT   N1   N2   N3   N4   N5   N6 ;
CARDS;
     3    0    1    1    0    1
     0    3    1    1    0    1
     1    1    4    1    0    1
     1    1    1    5    1    1
     0    0    0    1    2    1
     1    1    1    1    1    5
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS = 4   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6  ;
*;
*   E x W2 x E';
*;
DATA   CONDSPEC;
INPUT   N1   N2   N3   N4   N5   N6 ;
CARDS;
     4    0    2    1    0    1
     0    7    1    2    0    4
     2    1    7    2    0    2
     1    2    2   10    3    2
     0    0    0    3    5    2
     1    4    2    2    2   11
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS =  3   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6  ;
*;
*   E x W3 x E';
*;
DATA   CONDSPEC;
INPUT   N1   N2   N3   N4   N5   N6 ;
CARDS;
    20    0   10    5    0    5
     0   15    5    2    0    8
    10    5   27    6    0    6
     5    2    6   18    3    2
     0    0    0    3    5    2
     5    8    6    2    2   23
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS =  3   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6  ;
* i    =   5 ;
```

Appendix E. Repairman Example SAS Output File

```
                 SAS        20:44 FRIDAY, NOVEMBER 4, 1988  1

         OBS     N1     N2     N3     N4     N5     N6

          1      3      0      1      1      0      1
          2      0      3      1      1      0      1
          3      1      1      4      1      0      1
          4      1      1      1      5      1      1
          5      0      0      0      1      2      1
          6      1      1      1      1      1      5
```

```
INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                     PRIOR COMMUNALITY ESTIMATES: ONE

EIGENVALUES OF THE COVARIANCE MATRIX:  TOTAL = 10.2667  AVERAGE = 1.7111

                    1         2         3         4         5         6
EIGENVALUE   3.200000  2.724364  1.800000  1.742303  0.800000  0.000000
DIFFERENCE   0.475636  0.924364  0.057697  0.942303  0.800000
PROPORTION     0.3117    0.2654    0.1753    0.1697    0.0779    0.0000
CUMULATIVE     0.3117    0.5770    0.7524    0.9221    1.0000    1.0000

               4 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                              FACTOR PATTERN

                        FACTOR1   FACTOR2   FACTOR3   FACTOR4

               N1      -0.00000  -0.19707   0.86603   0.21094
               N2      -0.00000  -0.19707  -0.86603   0.21094
               N3      -0.00000  -0.77013  -0.00000   0.54749
               N4       0.77460   0.46214  -0.00000   0.43177
               N5       0.00000   0.75988  -0.00000  -0.35012
               N6      -0.77460   0.46214   0.00000   0.43177

                      VARIANCE EXPLAINED BY EACH FACTOR

                         FACTOR1   FACTOR2   FACTOR3   FACTOR4
              WEIGHTED   3.200000  2.724364  1.800000  1.742303
              UNWEIGHTED 1.200000  1.675350  1.500000  0.884174

               FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
        TOTAL COMMUNALITY: WEIGHTED =  9.466667   UNWEIGHTED =  5.259524

                   N1        N2        N3        N4        N5        N6
COMMUNALITY  0.833333  0.833333  0.892857  1.000000  0.700000  1.000000
WEIGHT       1.200000  1.200000  1.866667  2.666667  0.666667  2.666667
```

```
                            SAS      20:44 FRIDAY, NOVEMBER 4, 1988  3

ROTATION METHOD: VARIMAX

                        ORTHOGONAL TRANSFORMATION MATRIX

                           1          2          3          4

                 1    -0.00008   -0.00000   -0.70713    0.70709
                 2    -0.80002    0.00000    0.42428    0.42421
                 3     0.00000    1.00000    0.00000    0.00000
                 4     0.59997   -0.00000    0.56565    0.56575

                           ROTATED FACTOR PATTERN

                       FACTOR1    FACTOR2    FACTOR3    FACTOR4

             N1        0.28422    0.86603    0.03570    0.03574
             N2        0.28422   -0.86603    0.03570    0.03574
             N3        0.94461   -0.00000   -0.01706   -0.01695
             N4       -0.11074   -0.00000   -0.10743    0.98803
             N5       -0.81798   -0.00000    0.12435    0.12426
             N6       -0.11062   -0.00000    0.98804   -0.10739

                      VARIANCE EXPLAINED BY EACH FACTOR

                        FACTOR1    FACTOR2    FACTOR3    FACTOR4
             WEIGHTED   2.370860   1.800000   2.647972   2.647835
             UNWEIGHTED 1.747438   1.500000   1.006076   1.006010

                   FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
         TOTAL COMMUNALITY: WEIGHTED =  9.466667   UNWEIGHTED =  5.259524

                   N1         N2         N3         N4         N5         N6
COMMUNALITY     0.833333   0.833333   0.892857   1.000000   0.700000   1.000000
WEIGHT          1.200000   1.200000   1.866667   2.666667   0.666667   2.666667
```

```
OBS    N1    N2    N3    N4    N5    N6

 1      4     0     2     1     0     1
 2      0     7     1     2     0     4
 3      2     1     7     2     0     2
 4      1     2     2    10     3     2
 5      0     0     0     3     5     2
 6      1     4     2     2     2    11
```

```
                         SAS      20:44 FRIDAY, NOVEMBER 4, 1988  5

INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                    PRIOR COMMUNALITY ESTIMATES: ONE

EIGENVALUES OF THE COVARIANCE MATRIX:  TOTAL =  44.8  AVERAGE =  7.46667

                   1         2         3         4         5         6

EIGENVALUE     18.241880 12.982349  6.400000  5.389726  1.786045  0.000000
DIFFERENCE      5.259530  6.582349  1.010274  3.603680  1.786045
PROPORTION        0.4072    0.2898    0.1429    0.1203    0.0399    0.0000
CUMULATIVE        0.4072    0.6970    0.8398    0.9601    1.0000    1.0000

                3 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                              FACTOR PATTERN

                          FACTOR1   FACTOR2   FACTOR3

               N1        -0.33934  -0.63298   0.07329
               N2         0.72778   0.19465   0.39034
               N3        -0.16171  -0.53780   0.63777
               N4        -0.40546   0.82464   0.36485
               N5        -0.17743   0.71286  -0.56980
               N6         0.92711   0.16996  -0.01975

                    VARIANCE EXPLAINED BY EACH FACTOR

                           FACTOR1   FACTOR2   FACTOR3
              WEIGHTED   18.241880 12.982349  6.400000
              UNWEIGHTED  1.726389  1.944871  1.022672

                FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
        TOTAL COMMUNALITY: WEIGHTED = 37.624229   UNWEIGHTED =  4.693932

                   N1        N2        N3        N4        N5        N6
COMMUNALITY     0.521194  0.719921  0.722135  0.977550  0.864312  0.888819
WEIGHT          2.266667  7.466667  5.866667 11.066667  4.266667 13.866667
```

```
                        SAS      20:44 FRIDAY, NOVEMBER 4, 1988  6

ROTATION METHOD: VARIMAX

                        ORTHOGONAL TRANSFORMATION MATRIX

                                  1          2          3

                        1     -0.06883    0.92202   -0.38097
                        2     -0.60766    0.26412    0.74899
                        3      0.79121    0.28305    0.54210

                             ROTATED FACTOR PATTERN

                             FACTOR1    FACTOR2    FACTOR3

                    N1       0.46598   -0.45932   -0.30509
                    N2       0.14047    0.83293    0.08013
                    N3       0.84254   -0.11062    0.00454
                    N4      -0.18452   -0.05277    0.96991
                    N5      -0.87179   -0.13660    0.29263
                    N6      -0.18272    0.89412   -0.23661

                       VARIANCE EXPLAINED BY EACH FACTOR

                              FACTOR1    FACTOR2    FACTOR3
                   WEIGHTED    8.886661 16.926245 11.811323
                   UNWEIGHTED  1.774207  1.737870  1.181855

                 FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
          TOTAL COMMUNALITY: WEIGHTED = 37.624229   UNWEIGHTED =  4.693932

                     N1        N2        N3        N4        N5        N6
COMMUNALITY      0.521194  0.719921  0.722135  0.977550  0.864312  0.888819
WEIGHT           2.266667  7.466667  5.866667 11.066667  4.266667 13.866667
```

```
OBS    N1    N2    N3    N4    N5    N6

 1     20     0    10     5     0     5
 2      0    15     5     2     0     8
 3     10     5    27     6     0     6
 4      5     2     6    18     3     2
 5      0     0     0     3     5     2
 6      5     8     6     2     2    23
```

```
INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                      PRIOR COMMUNALITY ESTIMATES: ONE

EIGENVALUES OF THE COVARIANCE MATRIX:  TOTAL = 281.6  AVERAGE = 46.9333

                    1         2         3         4         5         6
EIGENVALUE    115.735 86.030914 42.193668 24.161913 13.478646  0.000000
DIFFERENCE  29.703945 43.837245 18.031756 10.683267 13.478646
PROPORTION     0.4110    0.3055    0.1498    0.0858    0.0479    0.0000
CUMULATIVE     0.4110    0.7165    0.8663    0.9521    1.0000    1.0000

                 3 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                                FACTOR PATTERN

                              FACTOR1   FACTOR2   FACTOR3

                    N1        0.78273   0.03569  -0.58222
                    N2       -0.39169   0.64646   0.46963
                    N3        0.86629   0.36788   0.32042
                    N4        0.27951  -0.62986   0.19649
                    N5       -0.53188  -0.55696  -0.06460
                    N6       -0.30376   0.84262  -0.28671

                       VARIANCE EXPLAINED BY EACH FACTOR

                               FACTOR1   FACTOR2   FACTOR3
                   WEIGHTED    115.735 86.030914 42.193668
                   UNWEIGHTED 1.969839  1.971451  0.787178

                 FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
        TOTAL COMMUNALITY: WEIGHTED =   243.959   UNWEIGHTED =  4.728468

                   N1        N2        N3        N4        N5        N6
COMMUNALITY  0.952913  0.791884  0.988469  0.513453  0.597271  0.884477
WEIGHT      56.666667 33.600000 88.000000 37.200000  4.266667 61.866667
```

```
                             SAS      20:44 FRIDAY, NOVEMBER 4, 1988  9
ROTATION METHOD: VARIMAX

                      ORTHOGONAL TRANSFORMATION MATRIX

                              1          2          3

                    1     0.79508   -0.33867    0.50314
                    2     0.49894    0.83689   -0.22511
                    3     0.34484   -0.43002   -0.83437

                        ROTATED FACTOR PATTERN

                          FACTOR1    FACTOR2    FACTOR3

               N1         0.43937    0.01515    0.87157
               N2         0.17307    0.47172   -0.73445
               N3         0.98281   -0.12330    0.08570
               N4        -0.02427   -0.70628    0.11848
               N5        -0.72305   -0.25820   -0.08834
               N6         0.08004    0.93134   -0.10330

                      VARIANCE EXPLAINED BY EACH FACTOR

                            FACTOR1    FACTOR2    FACTOR3
              WEIGHTED    99.595452  81.331629  63.032361
              UNWEIGHTED   1.718714   1.670849   1.338905

                 FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
          TOTAL COMMUNALITY: WEIGHTED =    243.959   UNWEIGHTED =   4.728468

                      N1         N2         N3         N4         N5         N6
COMMUNALITY     0.952913   0.791884   0.988469   0.513453   0.597271   0.884477
WEIGHT         56.666667  33.600000  88.000000  37.200000   4.266667  61.866667
```

Appendix F. Manufacturing Example Transition Specification

```
{operation completed for part i on machine j}
WHEN ALARM(op_com[i,j]);
        curr_op[i,j] :== done;
        opn_comp[i] :== opn_comp[i] + 1;

{part i to off-shuttle position k of machine j}
WHEN curr_op[i,j] = done & off_shu_pos[j,k] = 0;
        SET ALARM(mov_offshu[i,j,k], mach_off_time);
        mach_time_busy[j] :== mach_time_busy[j] + (system_time - mach_
                                start_time);
        SET ALARM(next_op[i], 0);

{part i in off-shuttle position k of machine j}
WHEN ALARM(mov_offshu[i,j,k]);
        off_shu_pos[j,k] :== i;
        mach_status[j] :== idle;
        SET ALARM(next_prt[j], 0);

{operations completed for part i}
WHEN ALARM(next_op[i] = tot_opn[part_type[i]] &
                         opn_comp[i] = tot_opn[part_type[i]]);
        part_time_sys[i] :== part_time_sys[i] + (system_time - part_
                              start_time[i]);
        DESTROY(part_type,i);
        CREATE(part_type,i);
        SET ALARM(next_op[i], 0);

{next operation for part i}
WHEN ALARM(next_op[i] < tot_opn[part_type[i]] & opn_comp[i] <
                              tot_opn[part_type[i]]);
        next_opn[i] :== olookup(i, opn_comp[i]);
        SET ALARM(next_stt[i,next_opn[i]], 0);

{next station for next operation for part i}
WHEN ALARM(next_stt[i,next_opn[i]]);
        next_stat[i] :== slookup(i, next_opn[i]);
        SET ALARM(find_cart[i], 0);

{find cart for part i}
WHEN ALARM(find_cart[i] = idle & cart_stat[j] = idle);
        cart_assign[i] :== j;
        SET ALARM(mov_cart[j,i,1], 0);

{move cart j to pickup part i}
WHEN ALARM(mov_cart[j,i,1]);
        cmov_time[j,i,1] :== mtime_calc[j,i];
        SET ALARM(cart_arr[j,i,1], cmov_time[j,i,1]);
```

```
{cart j arrives to pickup part i}
WHEN ALARM((cart_arr[j,i,1]) & off_shu_pos[curr_stat[i],1] = i);
        SET ALARM(to_cart[j,i], off_cart_time);

{part i arrives onto cart j}
WHEN ALARM(to_cart[j,i]);
        cart_stat[j] :== busy;
******* For all off-shu positions n;
        off_shu_pos[n-1] :== off_shu_pos[n];
        SET ALARM(mov_cart[j,i,2], 0);

{cart j arrives to drop off part i at machine k}
WHEN ALARM((cart_arr[j,i,2]) & on_shu_pos[next_stat[i],k] = 0);
        SET ALARM(to_onshu[j,i,next_stat[i],k], cart_onshu_mvtim
                         [next_stat[i]]);

{part i arrives into on-shuttle}
WHEN ALARM(to_onshu[j,i,next_stat[i],k]);
        on_shu_pos[next_stat[i],k] :== i;
        cart_stat[j] :== idle;
        SET ALARM(chk_mach_stat[next_stat[i],i], 0);
        curr_stat[i] :== next_stat[i];

{part i can move into machine}
WHEN ALARM((chk_mach_stat[j,i]) & on_shu_pos[j,1] = i & mach_stat[j] =
idle;
        SET ALARM(shu_mach[j,i], shu_mach_time);

{part i moves into machine}
WHEN ALARM(shu_mach[j,i]);
        For on-shu positions 2 to n;
        on_shu_pos[j,n-1] :== on_shu_pos[j,n];
        For last on-shu position n;
        on_shu_pos[j,n] :== 0;
        mach_stat[j] :== busy;
        mach_start_time :== system_time;
        SET ALARM(op_com[i,j], 0);

{next part from queue selected to move into machine}
WHEN ALARM((next_prt[j]) & on_shu_pos[j,1] = i & i = 0);
        SET ALARM(shu_mach[j,i], shu_mach_time);
```

NODE.CMP

NODE 1 operation completed for part i on machine j

CONTROL ATTRIBUTE	op_com[i,j]
OUTPUT ATTRIBUTE	curr_op[i,j]
INPUT ATTRIBUTE	opn_comp[i]
OUTPUT ATTRIBUTE	opn_comp[i]

NODE 2 part i to off-shuttle position k of machine j

CONTROL ATTRIBUTE	curr_op[i,j]
CONTROL ATTRIBUTE	off_shu_pos[j,k]
OUTPUT ATTRIBUTE	mov_offshu[i,j,k]
INPUT ATTRIBUTE	mach_off_time
INPUT ATTRIBUTE	mach_time_busy[j]
INPUT ATTRIBUTE	mach_start_time
OUTPUT ATTRIBUTE	mach_time_busy[j]
OUTPUT ATTRIBUTE	next_op[i]

NODE 3 part i in off-shuttle position k of machine j

CONTROL ATTRIBUTE	mov_offshu[i,j,k]
INPUT ATTRIBUTE	i
OUTPUT ATTRIBUTE	off_shu_pos[j,k]
OUTPUT ATTRIBUTE	mach_status[j]
OUTPUT ATTRIBUTE	next_prt[j]

NODE 4 operations completed for part i

CONTROL ATTRIBUTE	next_op[i]
INPUT ATTRIBUTE	tot_opn[part_type[i]]
CONTROL ATTRIBUTE	opn_comp[i]
INPUT ATTRIBUTE	tot_opn[part_type[i]]
INPUT ATTRIBUTE	part_time_sys[i]
INPUT ATTRIBUTE	part_start_time[i]
OUTPUT ATTRIBUTE	part_time_sys[i]
INPUT ATTRIBUTE	part_type
OUTPUT ATTRIBUTE	part_type
OUTPUT ATTRIBUTE	next_op[i]

NODE 5 next operation for part i

CONTROL ATTRIBUTE	next_op[i]
INPUT ATTRIBUTE	tot_opn[part_type[i]]
CONTROL ATTRIBUTE	opn_comp[i]

INPUT ATTRIBUTE	tot_opn[part_type[i]]
INPUT ATTRIBUTE	i
INPUT ATTRIBUTE	opn_comp[i]
OUTPUT ATTRIBUTE	next_opn[i]
OUTPUT ATTRIBUTE	next_stt[i,next_opn[i]]

NODE 6 next station for next operation for part i

CONTROL ATTRIBUTE	next_stt[i,next_opn[i]]
INPUT ATTRIBUTE	i
INPUT ATTRIBUTE	next_opn[i]
OUTPUT ATTRIBUTE	next_stat[i]
OUTPUT ATTRIBUTE	find_cart[i]

NODE 7 find cart for part i

CONTROL ATTRIBUTE	find_cart[i]
CONTROL ATTRIBUTE	cart_stat[j]
INPUT ATTRIBUTE	j
OUTPUT ATTRIBUTE	cart_assign[i]
OUTPUT ATTRIBUTE	mov_cart[j,i,1]

NODE 8 move cart j to pickup part i

CONTROL ATTRIBUTE	mov_cart[j,i,1]
INPUT ATTRIBUTE	mtime_calc[j,i]
OUTPUT ATTRIBUTE	cmov_time[j,i,1]
OUTPUT ATTRIBUTE	cart_arr[j,i,1]
INPUT ATTRIBUTE	cmov_time[j,i,1]

NODE 9 cart j arrives to pickup part i

CONTROL ATTRIBUTE	cart_arr[j,i,1]
CONTROL ATTRIBUTE	off_shu_pos[curr_stat[i],1]
INPUT ATTRIBUTE	i
OUTPUT ATTRIBUTE	to_cart[j,i]
INPUT ATTRIBUTE	off_cart_time

NODE 10 part i arrives onto cart j

CONTROL ATTRIBUTE	to_cart[j,i]
OUTPUT ATTRIBUTE	cart_stat[j]
INPUT ATTRIBUTE	off_shu_pos[n]
OUTPUT ATTRIBUTE	off_shu_pos[n-1]
OUTPUT ATTRIBUTE	mov_cart[j,i,2]

NODE 11   cart j arrives to drop off part i at machine k

CONTROL ATTRIBUTE	cart_arr[j,i,2]
CONTROL ATTRIBUTE	on_shu_pos[next_stat[i],k]
OUTPUT ATTRIBUTE	to_onshu[j,i,next_stat[i],k]
INPUT ATTRIBUTE	cart_onshu_mvtim [next_stat[i]]

NODE 12   part i arrives into on-shuttle

CONTROL ATTRIBUTE	to_onshu[j,i,next_stat[i],k]
INPUT ATTRIBUTE	i
OUTPUT ATTRIBUTE	on_shu_pos[next_stat[i],k]
OUTPUT ATTRIBUTE	cart_stat[j]
OUTPUT ATTRIBUTE	chk_mach_stat[next_stat[i],i]
INPUT ATTRIBUTE	next_stat[i]
OUTPUT ATTRIBUTE	curr_stat[i]

NODE 13   part i can move into machine

CONTROL ATTRIBUTE	chk_mach_stat[j,i]
CONTROL ATTRIBUTE	on_shu_pos[j,1]
INPUT ATTRIBUTE	i
CONTROL ATTRIBUTE	mach_stat[j]
OUTPUT ATTRIBUTE	shu_mach[j,i]
INPUT ATTRIBUTE	shu_mach_time

NODE 14   part i moves into machine

CONTROL ATTRIBUTE	shu_mach[j,i]
INPUT ATTRIBUTE	on_shu_pos[j,n]
OUTPUT ATTRIBUTE	on_shu_pos[j,n-1]
OUTPUT ATTRIBUTE	on_shu_pos[j,n]
OUTPUT ATTRIBUTE	mach_stat[j]
OUTPUT ATTRIBUTE	mach_start_time
OUTPUT ATTRIBUTE	op_com[i,j]

NODE 15   next part from queue selected to move into machine

CONTROL ATTRIBUTE	next_prt[j]
CONTROL ATTRIBUTE	on_shu_pos[j,1]
INPUT ATTRIBUTE	i
CONTROL ATTRIBUTE	i
OUTPUT ATTRIBUTE	shu_mach[j,i]
INPUT ATTRIBUTE	shu_mach_time

ERRORS.DAT

```
NON-MATCHING SUBSCRIPTS

NODE         ATTRIBUTE                        NODE          ATTRIBUTE

FATAL ERROR!!!  Subscripts are not compatible!  Results will be
inaccurate!!!
 2  off_shu_pos[j,k]                           10  off_shu_pos[n-1]

Warning:  Subscripts do not match exactly
 3  off_shu_pos[j,k]                            9  off_shu_pos[curr_stat[i],1]

FATAL ERROR!!!  Subscripts are not compatible!  Results will be
inaccurate!!!
 3  off_shu_pos[j,k]                           10  off_shu_pos[n]

Warning:  Subscripts do not match exactly
 8  mov_cart[j,i,1]                            10  mov_cart[j,i,2]

   *** The above edge has been disregarded ***

Warning:  Subscripts do not match exactly
 8  cart_arr[j,i,1]                            11  cart_arr[j,i,2]

   *** The above edge has been disregarded ***

FATAL ERROR!!!  Subscripts are not compatible!  Results will be
inaccurate!!!
 9  off_shu_pos[curr_stat[i],1]                10  off_shu_pos[n-1]

Warning:  Subscripts do not match exactly
11  on_shu_pos[next_stat[i],k]                 14  on_shu_pos[j,n-1]

Warning:  Subscripts do not match exactly
12  on_shu_pos[next_stat[i],k]                 13  on_shu_pos[j,1]

Warning:  Subscripts do not match exactly
12  on_shu_pos[next_stat[i],k]                 14  on_shu_pos[j,n]

Warning:  Subscripts do not match exactly
12  on_shu_pos[next_stat[i],k]                 15  on_shu_pos[j,1]

Warning:  Subscripts do not match exactly
12  chk_mach_stat[next_stat[i],i]              13  chk_mach_stat[j,i]

Warning:  Subscripts do not match exactly
13  on_shu_pos[j,1]                            14  on_shu_pos[j,n-1]

Warning:  Subscripts do not match exactly
14  on_shu_pos[j,n-1]                          15  on_shu_pos[j,1]
```

ATTRIBUTES WITHOUT A COUNTERPART

ATTRIBUTE	NODE	TYPE
mach_off_time	2	INPUT
mach_time_busy	2	INPUT
mach_time_busy	2	OUTPUT
i	3	INPUT
mach_status	3	OUTPUT
tot_opn	4	INPUT
part_time_sys	4	INPUT
part_start_time	4	INPUT
part_time_sys	4	OUTPUT
tot_opn	5	INPUT
i	5	INPUT
i	6	INPUT
j	7	INPUT
cart_assign	7	OUTPUT
mtime_calc	8	INPUT
cmov_time	8	INPUT
cmov_time	8	OUTPUT
i	9	INPUT
off_cart_time	9	INPUT
cart_onshu_mvtim	11	INPUT
i	12	INPUT
i	13	INPUT
shu_mach_time	13	INPUT
i	15	CONTROL
shu_mach_time	15	INPUT

<u>INF.DAT</u>

ATTRIBUTE		NODE	C/I/O	NODE	C/I/O	W2	W3
op_com	,	1	1	14	3	1	57
curr_op	,	1	3	2	1	1	57
opn_comp	,	1	3	4	1	1	19
opn_comp	,	1	3	5	1	1	19
off_shu_pos	,	2	1	3	3	1	3
mach_start_time	,	2	2	14	3	1	1
mov_offshu	,	2	3	3	1	2	60
next_op	,	2	3	4	1	1	19
next_op	,	2	3	5	1	1	19
off_shu_pos	,	3	3	9	1	1	19
next_prt	,	3	3	15	1	1	3
part_type	,	4	3	5	2	1	1
next_op	,	4	3	5	1	2	20
next_opn	,	5	3	6	2	1	19
next_stt	,	5	3	6	1	2	380
next_stat	,	6	3	11	2	1	1

```
next_stat         ,  6   3    12   2   1    19
find_cart         ,  6   3     7   1   1    19
cart_stat         ,  7   1    10   3   1     3
cart_stat         ,  7   1    12   3   1     3
mov_cart          ,  7   3     8   1   1    57
cart_arr          ,  8   3     9   1   1    57
curr_stat         ,  9   2    12   3   1     1
to_cart           ,  9   3    10   1   1    57
on_shu_pos        , 11   1    12   3   1    19
on_shu_pos        , 11   1    14   3   1    19
next_stat         , 11   2    12   3   2    20
to_onshu          , 11   3    12   1   3  1103
next_stat         , 11   3    12   2   4  1122
on_shu_pos        , 12   3    13   1   1     3
on_shu_pos        , 12   3    14   2   1    15
on_shu_pos        , 12   3    15   1   1     3
chk_mach_stat     , 12   3    13   1   2    60
on_shu_pos        , 13   1    14   3   1     3
mach_stat         , 13   1    14   3   2     6
shu_mach          , 13   3    14   1   3    63
shu_mach          , 14   1    15   3   1    57
on_shu_pos        , 14   3    15   1   2    60

NUMBER OF EDGES = 38
```

Appendix H. Manufacturing Example SAS Input File

```
OPTIONS LINESIZE=80;
*;
*   E x E';
*;
DATA   CONDSPEC;
INPUT N1  N2  N3  N4  N5  N6  N7  N8  N9  N10  N11  N12  N13  N14  N15 ;
CARDS;
    4   1   0   1   1    0    0    0    0    0    0    0    0    1    0
    1   5   1   1   1    0    0    0    0    0    0    0    0    1    0
    0   1   3   0   0    0    0    0    1    0    0    0    0    0    1
    1   1   0   3   1    0    0    0    0    0    0    0    0    0    0
    1   1   0   1   4    1    0    0    0    0    0    0    0    0    0
    0   0   0   0   1    4    1    0    0    0    1    1    0    0    0
    0   0   0   0   0    1    4    1    0    1    0    1    0    0    0
    0   0   0   0   0    0    1    2    1    0    0    0    0    0    0
    0   0   1   0   0    0    0    1    4    1    0    1    0    0    0
    0   0   0   0   0    0    1    0    1    2    0    0    0    0    0
    0   0   0   0   0    1    0    0    0    0    3    1    0    1    0
    0   0   0   0   0    1    1    0    1    0    1    7    1    1    1
    0   0   0   0   0    0    0    0    0    0    0    1    2    1    0
    1   1   0   0   0    0    0    0    0    0    1    1    1    6    1
    0   0   1   0   0    0    0    0    0    0    0    1    0    1    3
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS =  6   ROTATE=VARIMAX  ;
   VAR     N1   N2   N3   N4   N5   N6   N7   N8   N9  N10  N11  N12
N13  N14  N15  ;
*;
*   E x W2 x E';
*;
DATA   CONDSPEC;
INPUT N1  N2  N3  N4  N5  N6  N7  N8  N9  N10  N11  N12  N13  N14  N15 ;
CARDS;
    4   1   0   1   1    0    0    0    0    0    0    0    0    1    0
    1   6   2   1   1    0    0    0    0    0    0    0    0    1    0
    0   2   4   0   0    0    0    0    1    0    0    0    0    0    1
    1   1   0   4   2    0    0    0    0    0    0    0    0    0    0
    1   1   0   2   6    2    0    0    0    0    0    0    0    0    0
    0   0   0   0   2    5    1    0    0    0    1    1    0    0    0
    0   0   0   0   0    1    4    1    0    1    0    1    0    0    0
    0   0   0   0   0    0    1    2    1    0    0    0    0    0    0
    0   0   1   0   0    0    0    1    4    1    0    1    0    0    0
    0   0   0   0   0    0    1    0    1    2    0    0    0    0    0
    0   0   0   0   0    1    0    0    0    0    6    4    0    1    0
    0   0   0   0   0    1    1    0    1    0    4   11    2    1    1
    0   0   0   0   0    0    0    0    0    0    0    2    5    3    0
    1   1   0   0   0    0    0    0    0    0    1    1    3    9    2
    0   0   1   0   0    0    0    0    0    0    0    1    0    2    4
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS =  6   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6   N7   N8   N9  N10  N11  N12
N13  N14  N15  ;
*;
*   E x W3 x E';
*;
DATA   CONDSPEC;
INPUT N1  N2  N3  N4  N5  N6  N7  N8  N9  N10  N11  N12  N13  N14  N15 ;
CARDS;
  152  57   0  19  19    0    0    0    0    0    0    0    0   57    0
   57 156  60  19  19    0    0    0    0    0    0    0    0    1    0
    0  60  82   0   0    0    0    0   19    0    0    0    0    0    3
   19  19   0  58  20    0    0    0    0    0    0    0    0    0    0
   19  19   0  20 438  380    0    0    0    0    0    0    0    0    0
    0   0   0   0 380  419   19    0    0    0    1   19    0    0    0
    0   0   0   0   0   19   82   57    0    3    0    3    0    0    0
    0   0   0   0   0    0   57  114   57    0    0    0    0    0    0
    0   0  19   0   0    0    0   57  134   57    0    1    0    0    0
    0   0   0   0   0    0    3    0   57   60    0    0    0    0    0
    0   0   0   0   0    1    0    0    0    0 1142 1122    0   19    0
    0   0   0   0   0   19    3    0    1    0 1122 1223   60   15    3
    0   0   0   0   0    0    0    0    0    0    0   60  123   63    0
   57   1   0   0   0    0    0    0    0    0   19   15   63  215   60
    0   0   3   0   0    0    0    0    0    0    0    3    0   60   66
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS =  7   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6   N7   N8   N9  N10  N11  N12
N13  N14  N15  ;
* i    =  19 ;
* j    =   3 ;
* k    =   1 ;
* n    =   5 ;
* n-1  =   4 ;
```

Appendix I. Manufacturing Example SAS Output File

```
                                    SAS       10:25 THURSDAY, NOVEMBER 17, 1988  1

OBS  N1  N2  N3  N4  N5  N6  N7  N8  N9  N10  N11  N12  N13  N14  N15

  1   4   1   0   1   1   0   0   0   0    0    0    0    0    1    0
  2   1   5   1   1   1   0   0   0   0    0    0    0    0    1    0
  3   0   1   3   0   0   0   0   0   1    0    0    0    0    0    1
  4   1   1   0   3   1   0   0   0   0    0    0    0    0    0    0
  5   1   1   0   1   4   1   0   0   0    0    0    0    0    0    0
  6   0   0   0   0   1   4   1   0   0    0    1    1    0    0    0
  7   0   0   0   0   0   1   4   1   0    1    0    1    0    0    0
  8   0   0   0   0   0   0   1   2   1    0    0    0    0    0    0
  9   0   0   1   0   0   0   0   1   4    1    0    1    0    0    0
 10   0   0   0   0   0   0   1   0   1    2    0    0    0    0    0
 11   0   0   0   0   0   1   0   0   0    0    3    1    0    1    0
 12   0   0   0   0   0   1   1   0   1    0    1    7    1    1    1
 13   0   0   0   0   0   0   0   0   0    0    0    1    2    1    0
 14   1   1   0   0   0   0   0   0   0    0    1    1    1    6    1
 15   0   0   1   0   0   0   0   0   0    0    0    1    0    1    3
```

```
                    SAS       10:25 THURSDAY, NOVEMBER 17, 1988           2

INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                 PRIOR COMMUNALITY ESTIMATES: ONE

                    EIGENVALUES OF THE COVARIANCE MATRIX
                  TOTAL =    16.4667    AVERAGE =     1.09778

                        1          2          3          4          5
   EIGENVALUE    4.323441   3.234584   2.129941   1.628266   1.115005
   DIFFERENCE    1.088857   1.104643   0.501675   0.513261   0.050260
   PROPORTION      0.2626     0.1964     0.1293     0.0989     0.0677
   CUMULATIVE      0.2626     0.4590     0.5883     0.6872     0.7549

                        6          7          8          9         10
   EIGENVALUE    1.064745   0.831503   0.571358   0.515635   0.322260
   DIFFERENCE    0.233242   0.260145   0.055723   0.193376   0.036545
   PROPORTION      0.0647     0.0505     0.0347     0.0313     0.0196
   CUMULATIVE      0.8196     0.8701     0.9048     0.9361     0.9557

                       11         12         13         14         15
   EIGENVALUE    0.285714   0.254303   0.169131   0.020781   0.000000
   DIFFERENCE    0.031412   0.085172   0.148350   0.020781
   PROPORTION      0.0174     0.0154     0.0103     0.0013     0.0000
   CUMULATIVE      0.9730     0.9885     0.9987     1.0000     1.0000

               6 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                              FACTOR PATTERN

               FACTOR1    FACTOR2    FACTOR3    FACTOR4    FACTOR5    FACTOR6

 N1           -0.52781    0.38475    0.25344    0.08427    0.42059   -0.28469
 N2           -0.59838    0.36968    0.07619    0.42346    0.00595    0.50793
 N3           -0.18124   -0.09971   -0.54514    0.41027   -0.47303    0.25012
 N4           -0.57871    0.07059    0.34237    0.30099    0.22851   -0.16163
 N5           -0.50586   -0.03318    0.62535    0.19318   -0.07910   -0.22910
 N6            0.29372   -0.24227    0.68342   -0.22922   -0.38791    0.07974
 N7            0.34781   -0.44891    0.19375   -0.35889    0.42055    0.52575
 N8            0.08814   -0.48495   -0.31566   -0.19450    0.35990    0.07980
 N9            0.23090   -0.39245   -0.56946    0.34217    0.11025   -0.27821
 N10           0.08814   -0.48495   -0.31566   -0.19450    0.35990    0.07980
 N11           0.40944    0.28020    0.25879   -0.27003   -0.38629   -0.10703
 N12           0.86746    0.19397    0.21169    0.38859    0.11340    0.01424
 N13           0.41611    0.46137   -0.05335   -0.03141    0.13250   -0.09578
 N14           0.12105    0.90072   -0.17072   -0.30581    0.05523    0.01553
 N15           0.29091    0.33273   -0.32154    0.12107   -0.36642    C.05847
```

```
                          SAS      10:25 THURSDAY, NOVEMBER 17, 1988    3

                        VARIANCE EXPLAINED BY EACH FACTOR

             FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6
WEIGHTED     4.323441  3.234584  2.129941  1.628266  1.115005  1.064745
UNWEIGHTED   2.728954  2.436301  2.150841  1.189819  1.374657  0.877623


INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                 FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
   TOTAL COMMUNALITY: WEIGHTED = 13.495981   UNWEIGHTED = 10.758195

                       N1        N2        N3        N4        N5
   COMMUNALITY   0.755891  0.937880  0.794609  0.626049  0.744124
   WEIGHT        1.123810  1.666667  0.685714  0.685714  1.123810

                       N6        N7        N8        N9       N10
   COMMUNALITY   0.821404  0.942115  0.516310  0.738249  0.516310
   WEIGHT        1.123810  1.123810  0.352381  1.123810  0.352381

                      N11       N12       N13       N14       N15
   COMMUNALITY   0.546707  0.998992  0.416572  0.951917  0.451065
   WEIGHT        0.685714  3.066667  0.352381  2.314286  0.685714

ROTATION METHOD: VARIMAX

                       ORTHOGONAL TRANSFORMATION MATRIX

                  1         2         3         4         5         6

     1     -0.47627   0.18182   0.14979   0.16450  -0.39327   0.73209
     2      0.10481  -0.46684   0.06642   0.81930   0.27757   0.13555
     3      0.53898  -0.05628   0.75994  -0.22185   0.03827   0.27954
     4      0.08322  -0.38974  -0.43607  -0.42593   0.40450   0.55316
     5      0.57220   0.63211  -0.37395   0.26625   0.02961   0.24785
     6     -0.37060   0.44089   0.25622   0.01311   0.77610   0.01094
```

```
                      SAS     10:25 THURSDAY, NOVEMBER 17, 1988        4

                          ROTATED FACTOR PATTERN

              FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6

N1            0.78149  -0.18235  -0.12787   0.24453   0.14965  -0.11566
N2            0.21521  -0.22300  -0.06393   0.01542   0.90653  -0.12538
N3           -0.54718  -0.30435  -0.38598  -0.28798   0.36880  -0.18615
N4            0.68326  -0.20157  -0.07993  -0.18280   0.26336  -0.09703
N5            0.63022  -0.33798   0.28390  -0.35548   0.11166  -0.11528
N6           -0.06752   0.00733   0.81271  -0.30639  -0.19892   0.15117
N7           -0.09235   0.89941   0.30347  -0.08182   0.02134   0.15940
N8           -0.10277   0.59867  -0.28821  -0.13307  -0.18744  -0.10696
N9           -0.26337   0.07091  -0.68596  -0.27725  -0.29578   0.17021
N10          -0.10277   0.59867  -0.28821  -0.13307  -0.18744  -0.10696
N11          -0.22999  -0.25705   0.51139   0.25026  -0.27706   0.16379
N12          -0.18677  -0.01823   0.09548   0.11953  -0.10761   0.96374
N13          -0.06988  -0.08296  -0.00796   0.50568  -0.12074   0.36667
N14          -0.05487  -0.22793   0.06490   0.94091   0.08586   0.00769
N15          -0.49824  -0.33736  -0.07947   0.24343   0.04915   0.14498

                     VARIANCE EXPLAINED BY EACH FACTOR

               FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6
WEIGHTED      2.157557  1.754417  1.876781  2.767642  1.829716  3.109869
UNWEIGHTED    2.267939  2.100347  1.926875  1.792805  1.372511  1.297717

             FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
      TOTAL COMMUNALITY: WEIGHTED = 13.495981   UNWEIGHTED = 10.758195

                        N1         N2         N3         N4         N5
    COMMUNALITY   0.755891   0.937880   0.794609   0.626049   0.744124
    WEIGHT        1.123810   1.666667   0.685714   0.685714   1.123810

                        N6         N7         N8         N9        N10
    COMMUNALITY   0.821404   0.942115   0.516310   0.738249   0.516310
    WEIGHT        1.123810   1.123810   0.352381   1.123810   0.352381

                       N11        N12        N13        N14        N15
    COMMUNALITY   0.546707   0.998992   0.416572   0.951917   0.451065
    WEIGHT        0.685714   3.066667   0.352381   2.314286   0.685714
```

```
                        SAS     10:25 THURSDAY, NOVEMBER 17, 1988          5

OBS  N1  N2  N3  N4  N5  N6  N7  N8  N9  N10  N11  N12  N13  N14  N15

  1   4   1   0   1   1   0   0   0   0   0    0     0    0    1    0
  2   1   6   2   1   1   0   0   0   0   0    0     0    0    1    0
  3   0   2   4   0   0   0   0   0   1   0    0     0    0    0    1
  4   1   1   0   4   2   0   0   0   0   0    0     0    0    0    0
  5   1   1   0   2   6   2   0   0   0   0    0     0    0    0    0
  6   0   0   0   0   2   5   1   0   0   0    1     1    0    0    0
  7   0   0   0   0   0   1   4   1   0   1    0     1    0    0    0
  8   0   0   0   0   0   0   1   2   1   0    0     0    0    0    0
  9   0   0   1   0   0   0   0   1   4   1    0     1    0    0    0
 10   0   0   0   0   0   0   1   0   1   2    0     0    0    0    0
 11   0   0   0   0   0   1   0   0   0   0    6     4    0    1    0
 12   0   0   0   0   0   1   1   0   1   0    4    11    2    1    1
 13   0   0   0   0   0   0   0   0   0   0    0     2    5    3    0
 14   1   1   0   0   0   0   0   0   0   0    1     1    3    9    2
 15   0   0   1   0   0   0   0   0   0   0    0     1    0    2    4
```

```
                    SAS      10:25 THURSDAY, NOVEMBER 17, 1988        6

INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

              PRIOR COMMUNALITY ESTIMATES: ONE

                 EIGENVALUES OF THE COVARIANCE MATRIX
               TOTAL =   33.7333   AVERAGE =    2.24889

                      1         2         3         4         5
    EIGENVALUE  11.492141  7.199870  3.877136  3.104295  1.784399
    DIFFERENCE   4.292272  3.322734  0.772841  1.319896  0.284862
    PROPORTION     0.3407    0.2134    0.1149    0.0920    0.0529
    CUMULATIVE     0.3407    0.5541    0.6690    0.7611    0.8140

                      6         7         8         9        10
    EIGENVALUE   1.499537  1.367259  1.121034  0.768699  0.639928
    DIFFERENCE   0.132278  0.246224  0.352335  0.128771  0.324170
    PROPORTION     0.0445    0.0405    0.0332    0.0228    0.0190
    CUMULATIVE     0.8584    0.8990    0.9322    0.9550    0.9739

                     11        12        13        14        15
    EIGENVALUE   0.315758  0.285714  0.249753  0.027811  0.000000
    DIFFERENCE   0.030044  0.035961  0.221941  0.027811
    PROPORTION     0.0094    0.0085    0.0074    0.0008    0.0000
    CUMULATIVE     0.9833    0.9918    0.9992    1.0000    1.0000

           6 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                          FACTOR PATTERN

           FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6

N1        -0.30178   0.19888   0.41712   0.17228  -0.25038  -0.42386
N2        -0.37059   0.17186   0.20439   0.79669   0.06801   0.21580
N3        -0.28584   0.03628  -0.34932   0.68119   0.24160   0.31144
N4        -0.40986  -0.14151   0.57622   0.16384  -0.39802  -0.25059
N5        -0.38940  -0.27218   0.79694  -0.10337  -0.07484   0.12952
N6         0.07775  -0.43033   0.46265  -0.39817   0.45974   0.36724
N7         0.07306  -0.32088  -0.29345  -0.42509   0.07547   0.23276
N8        -0.15844  -0.19421  -0.52256  -0.29545  -0.12322  -0.05273
N9         0.02531  -0.22474  -0.58020   0.04721  -0.25229  -0.00426
N10       -0.15844  -0.19421  -0.52256  -0.29545  -0.12322  -0.05273
N11        0.79085  -0.23104   0.17755   0.13158   0.31046  -0.32735
N12        0.94480  -0.22765   0.06261   0.13461  -0.11978   0.07836
N13        0.48696   0.60981   0.08549  -0.16722  -0.37876   0.32421
N14        0.29763   0.92097   0.15021  -0.08182   0.10387  -0.02992
N15        0.21226   0.46834  -0.14424   0.07229   0.33262   0.01790
```

```
                  SAS      10:25 THURSDAY, NOVEMBER 17, 1988          7

                    VARIANCE EXPLAINED BY EACH FACTOR

             FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6
WEIGHTED    11.492141  7.199870  3.877136  3.104295  1.784399  1.499537
UNWEIGHTED   2.580833  2.123196  2.573990  1.757311  0.976396  0.817046

INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

           FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
   TOTAL COMMUNALITY: WEIGHTED = 28.957376   UNWEIGHTED = 10.828772

                      N1        N2        N3        N4        N5
    COMMUNALITY  0.576644  0.894549  0 824425  0.768100  0.893895
    WEIGHT       1.123810  2.457143  1.266667  1.266667  2.600000

                      N6        N7        N8        N9       N10
    COMMUNALITY  0.910053  0.434985  0.441152  0.453684  0.441152
    WEIGHT       1.809524  1.123810  0.352381  1.123810  0.352381

                     N11       N12       N13       N14       N15
    COMMUNALITY  0.931208  0.986998  0.892841  0.977709  0.401379
    WEIGHT       3.171429  8.123810  2.238095  5.457143  1.266667

ROTATION METHOD: VARIMAX

                    ORTHOGONAL TRANSFORMATION MATRIX

                 1         2         3         4         5         6

     1    -0.00018   0.85938   0.24732  -0.18651  -0.20622   0.35070
     2    -0.14255  -0.32604   0.71657   0.34453   0.02347   0.49056
     3    -0.87804   0.08066  -0.33137   0.28149  -0.15802   0.09236
     4     0.00024   0.32637  -0.01676   0.39293   0.85622  -0.07547
     5    -0.41042   0.03906   0.53819  -0.38946   0.12057  -0.61169
     6    -0.20070  -0.20156  -0.16008  -0.67912   0.42930   0.49796
```

```
                         SAS      10:25 THURSDAY, NOVEMBER 17, 1988          8

                              ROTATED FACTOR PATTERN

              FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6

    N1       -0.20667  -0.15866  -0.14013   0.69528  -0.06365  -0.04066
    N2       -0.27493  -0.13885  -0.04752   0.32586   0.83114  -0.02105
    N3        0.14010  -0.11666   0.13981  -0.07045   0.86108  -0.15882
    N4       -0.27201  -0.17118  -0.57055   0.57946  -0.02514  -0.05362
    N5       -0.65618  -0.24439  -0.61470   0.10375  -0.09395  -0.07841
    N6       -0.60739   0.05842  -0.24713  -0.61744  -0.22708  -0.20941
    N7        0.22560  -0.03896  -0.10414  -0.56127  -0.23117  -0.05707
    N8        0.54763  -0.20560  -0.05812  -0.21674  -0.17977  -0.12769
    N9        0.64589   0.05464  -0.09841  -0.12577   0.08937  -0.00632
    N10       0.54763  -0.20560  -0.05812  -0.21674  -0.17977  -0.12769
    N11      -0.18479   0.89034   0.18848  -0.02403  -0.18701  -0.18243
    N12       0.01077   0.91467  -0.02947  -0.19070  -0.07562   0.32759
    N13      -0.07174   0.09184   0.27614   0.00496  -0.14928   0.88357
    N14      -0.29988  -0.04899   0.74584   0.25179  -0.13387   0.49778
    N15      -0.08024   0.05106   0.61082  -0.03213   0.09970   0.09086

                         VARIANCE EXPLAINED BY EACH FACTOR

               FACTOR1   FACTOR2   FACTOR3   FACTOR4   FACTOR5   FACTOR6
   WEIGHTED   3.496379  9.672114  5.381773  3.003144  3.167626  4.236341
   UNWEIGHTED 2.215042  1.882314  1.875214  1.848540  1.714230  1.293432

                  FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
      TOTAL COMMUNALITY: WEIGHTED = 28.957376   UNWEIGHTED = 10.828772

                          N1        N2        N3        N4        N5
       COMMUNALITY  0.576644  0.894549  0.824425  0.768100  0.893895
       WEIGHT       1.123810  2.457143  1.266667  1.266667  2.600000

                          N6        N7        N8        N9       N10
       COMMUNALITY  0.910053  0.434985  0.441152  0.453684  0.441152
       WEIGHT       1.809524  1.123810  0.352381  1.123810  0.352381

                         N11       N12       N13       N14       N15
       COMMUNALITY  0.931208  0.986998  0.892841  0.977709  0.401379
       WEIGHT       3.171429  8.123810  2.238095  5.457143  1.266667
```

```
                         SAS       10:25 THURSDAY, NOVEMBER 17, 1988         9

OBS N1    N2 N3 N4   N5   N6 N7   N8   N9 N10   N11   N12 N13 N14 N15

  1 152   57  0 19   19    0  0    0    0   0     0     0   0  57   0
  2  57  156 60 19   19    0  0    0    0   0     0     0   0   1   0
  3   0   60 82  0    0    0  0    0   19   0     0     0   0   0   3
  4  19   19  0 58   20    0  0    0    0   0     0     0   0   0   0
  5  19   19  0 20  438  380  0    0    0   0     0     0   0   0   0
  6   0    0  0  0  380  419 19    0    0   0     1    19   0   0   0
  7   0    0  0  0    0   19 82   57    0   3     0     3   0   0   0
  8   0    0  0  0    0    0 57  114   57   0     0     0   0   0   0
  9   0    0 19  0    0    0  0   57  134  57     0     1   0   0   0
 10   0    0  0  0    0    0  3    0   57  60     0     0   0   0   0
 11   0    0  0  0    0    1  0    0    0   0  1142  1122   0  19   0
 12   0    0  0  0    0   19  3    0    1   0  1122  1223  60  15   3
 13   0    0  0  0    0    0  0    0    0   0     0    60 123  63   0
 14  57    1  0  0    0    0  0    0    0   0    19    15  63 215  60
 15   0    0  3  0    0    0  0    0    0   0     0     3   0  60  66
```

```
                   SAS      10:25 THURSDAY, NOVEMBER 17, 1988       10

INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

              PRIOR COMMUNALITY ESTIMATES: ONE

                 EIGENVALUES OF THE COVARIANCE MATRIX
               TOTAL =    379931   AVERAGE =    25328.7

                    1         2         3         4         5
EIGENVALUE     327830   39240.1   4653.97    3251.5   1389.33
DIFFERENCE     288589   34586.1   1402.47   1862.17   203.735
PROPORTION     0.8629    0.1033    0.0122    0.0086    0.0037
CUMULATIVE     0.8629    0.9661    0.9784    0.9870    0.9906

                    6         7         8         9        10
EIGENVALUE    1185.59   888.862   744.777   289.529   173.706
DIFFERENCE    296.733   144.085   455.249   115.822 41.763777
PROPORTION     0.0031    0.0023    0.0020    0.0008    0.0005
CUMULATIVE     0.9937    0.9961    0.9980    0.9988    0.9993

                   11        12        13        14        15
EIGENVALUE    131.943 90.105628 55.008382  6.696376  0.000000
DIFFERENCE  41.837076 35.097246 48.312006  6.696376
PROPORTION     0.0003    0.0002    0.0001    0.0000    0.0000
CUMULATIVE     0.9996    0.9998    1.0000    1.0000    1.0000

           7 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                           FACTOR PATTERN

                   FACTOR1   FACTOR2   FACTOR3   FACTOR4

           N1     -0.19877  -0.13432   0.48148   0.57186
           N2     -0.20292  -0.12832  -0.02040   0.90593
           N3     -0.17795  -0.20742  -0.23014   0.57578
           N4     -0.20330   0.05244   0.00065   0.46980
           N5     -0.18529   0.97969   0.02225   0.02766
           N6     -0.15126   0.98542   0.00832  -0.03640
           N7     -0.15648  -0.04137  -0.37989  -0.33006
           N8     -0.18335  -0.21959  -0.53517  -0.42400
           N9     -0.18757  -0.24079  -0.56378  -0.36333
           N10    -0.15900  -0.19680  -0.42700  -0.28972
           N11     0.99920   0.01775  -0.00492   0.00842
           N12     0.99927   0.02109   0.00468  -0.00557
           N13     0.18058  -0.16843   0.56240  -0.34228
           N14    -0.06707  -0.24514   0.90319  -0.23811
           N15    -0.12555  -0.20739   0.62719  -0.26956
```

```
                    SAS      10:25 THURSDAY, NOVEMBER 17, 1988       11

        INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                               FACTOR PATTERN

                          FACTOR5    FACTOR6    FACTOR7

               N1         0.48150   -0.05756    0.34620
               N2        -0.10483    0.04619    0.05145
               N3        -0.31322    0.25739   -0.21500
               N4         0.02493   -0.10222    0.03281
               N5         0.01882    0.01837    0.00964
               N6        -0.00566   -0.00707   -0.00083
               N7         0.12181   -0.76069    0.05687
               N8         0.32605   -0.35581    0.22584
               N9         0.30511    0.54148    0.19610
               N10        0.20166    0.66573    0.09108
               N11        0.02301    0.00074   -0.01930
               N12       -0.01867   -0.00001    0.01876
               N13       -0.54089    0.05579    0.43959
               N14        0.15721    0.06314   -0.06602
               N15        0.10366    0.05253   -0.53021

                     VARIANCE EXPLAINED BY EACH FACTOR

                           FACTOR1    FACTOR2    FACTOR3    FACTOR4
              WEIGHTED      327830    39240.1    4653.97     3251.5
              UNWEIGHTED  2.379254   2.290014   2.742125   2.453347

                           FACTOR5    FACTOR6    FACTOR7
              WEIGHTED     1389.33    1185.59    888.862
              UNWEIGHTED  0.925751   1.534032   0.750323

                  FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
          TOTAL COMMUNALITY: WEIGHTED =     378439   UNWEIGHTED = 13.074847

                              N1         N2         N3         N4         N5
          COMMUNALITY   0.971407   0.894545   0.669756   0.276938   0.996177
          WEIGHT         1725.92     1815.6    635.781    256.352    20443.4

                              N6         N7         N8         N9        N10
          COMMUNALITY   0.995415   0.876172   0.831927   0.967773   0.822432
          WEIGHT           19562    611.352    1144.89    1430.55    421.286

                             N11        N12        N13        N14        N15
          COMMUNALITY   0.999704   0.999733   0.983341   0.970099   0.819428
          WEIGHT          158259     168568    1333.11    3235.95      486.6
```

```
                    SAS      10:25 THURSDAY, NOVEMBER 17, 1988        12
ROTATION METHOD: VARIMAX

                   ORTHOGONAL TRANSFORMATION MATRIX

                       1          2          3          4

           1     0.97862   -0.10489   -0.04100   -0.10351
           2     0.07795    0.97525   -0.02533   -0.13565
           3     0.01545    0.08443   -0.21935   -0.36267
           4    -0.01426   -0.09429   -0.55574   -0.35079
           5     0.17838    0.09466    0.39333    0.36092
           6     0.05876    0.09909   -0.65129    0.73411
           7     0.02281    0.05562    0.24864    0.21709

                       5          6          7

           1    -0.06113   -0.10623    0.06240
           2    -0.12170   -0.07018   -0.06353
           3     0.71039    0.34420    0.43574
           4    -0.38745    0.53772   -0.34603
           5     0.31704    0.57280   -0.49561
           6     0.11527   -0.02960    0.09732
           7    -0.46139    0.49719    0.65373

                       ROTATED FACTOR PATTERN

                  FACTOR1    FACTOR2    FACTOR3    FACTOR4

      N1         -0.11530   -0.06429   -0.09892   -0.12975
      N2         -0.23663   -0.19348   -0.54594   -0.26475
      N3         -0.24773   -0.27344   -0.60125   -0.04273
      N4         -0.20236    0.02228   -0.16969   -0.21003
      N5         -0.10035    0.97829   -0.03964   -0.10912
      N6         -0.07201    0.97975    0.00182   -0.11568
      N7         -0.17920   -0.08557    0.83170   -0.22675
      N8         -0.15636   -0.19196    0.78224    0.29709
      N9         -0.11514   -0.13505    0.15548    0.93420
      N10        -0.09623   -0.09386   -0.06544    0.88092
      N11         0.98272   -0.08753   -0.04125   -0.10235
      N12         0.97679   -0.08405   -0.04211   -0.10871
      N13         0.09398   -0.12467   -0.07606   -0.13857
      N14        -0.03714   -0.11587   -0.05252   -0.11508
      N15        -0.11601   -0.12520   -0.10264   -0.13090
```

```
                     SAS     10:25 THURSDAY, NOVEMBER 17, 1988        13

ROTATION METHOD: VARIMAX

                        ROTATED FACTOR PATTERN

                     FACTOR5    FACTOR6    FACTOR7

          N1         0.13526    0.95340   -0.00987
          N2        -0.38913    0.47484   -0.23679
          N3        -0.32089   -0.03007   -0.25772
          N4        -0.19453    0.30438   -0.17916
          N5        -0.09918   -0.01151   -0.07492
          N6        -0.09289   -0.07325   -0.05426
          N7        -0.20270   -0.16815   -0.15568
          N8        -0.21981   -0.06773   -0.13255
          N9        -0.15029   -0.09636   -0.08666
          N10       -0.05876   -0.13096   -0.05884
          N11       -0.05371   -0.10100    0.03222
          N12       -0.07274   -0.11039    0.08649
          N13        0.17372   -0.09076    0.94634
          N14        0.85538    0.26252    0.37241
          N15        0.86646   -0.10697   -0.02096

                  VARIANCE EXPLAINED BY EACH FACTOR

                      FACTOR1    FACTOR2    FACTOR3    FACTOR4
         WEIGHTED      314247    41017.1    2577.23    6108.56
         UNWEIGHTED  2.209478   2.164373   2.054335   2.018439

                      FACTOR5    FACTOR6    FACTOR7
         WEIGHTED     4987.52    6061.13    3440.07
         UNWEIGHTED  1.965219   1.403774   1.259229

          FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
   TOTAL COMMUNALITY: WEIGHTED =     378439   UNWEIGHTED = 13.074847

                        N1         N2         N3         N4         N5
    COMMUNALITY   0.971407   0.894545   0.669756   0.276938   0.996177
    WEIGHT         1725.92     1815.6    635.781    256.352    20443.4

                        N6         N7         N8         N9        N10
    COMMUNALITY   0.995415   0.876172   0.831927   0.967773   0.822432
    WEIGHT           19562    611.352    1144.89    1430.55    421.286

                       N11        N12        N13        N14        N15
    COMMUNALITY   0.999704   0.999733   0.983341   0.970099   0.819428
    WEIGHT          158259     168568    1333.11    3235.95      486.6
```

Appendix J. Time Based Example SAS Input

```
OPTIONS LINESIZE=80;
*;
*   E x TIMEBASED WEIGHT x E';
*;
DATA   CONDSPEC;
INPUT   N1   N2   N3   N4   N5   N6 ;
CARDS;
           150.0001        0 50.00011       50        0       50
                  0 132.2815   33.516       20        0  78.7655
           50.00011   33.516 203.5161       60        0       60
                 50       20       60 179.7531  29.7531       20
                  0        0        0  29.7531  49.7531       20
                 50  78.7655       60       20       20 228.7655
;
PROC PRINT;
PROC FACTOR  COV  NFACTORS = 3   ROTATE=VARIMAX  ;
   VAR    N1   N2   N3   N4   N5   N6  ;
* i    =   5 ;
* 10 x EXP(-.025 x T) ;
```

Appendix K. Time Based Example SAS Output File

```
                         SAS    19:15 TUESDAY, NOVEMBER 15, 1988  1

OBS    N1        N2        N3        N4        N5         N6

 1    150     0.000    50.000    50.000    0.0000     50.000
 2      0   132.282    33.516    20.000    0.0000     78.766
 3     50    33.516   203.516    60.000    0.0000     60.000
 4     50    20.000    60.000   179.753   29.7531     20.000
 5      0     0.000     0.000    29.753   49.7531     20.000
 6     50    78.766    60.000    20.000   20.0000    228.766
```

```
INITIAL FACTOR METHOD: PRINCIPAL COMPONENTS

                    PRIOR COMMUNALITY ESTIMATES: ONE

EIGENVALUES OF THE COVARIANCE MATRIX:  TOTAL = 20880.9 AVERAGE = 3480.14

                  1         2         3         4         5         6
EIGENVALUE     8596.65   5476.26   3335.78   2277.38   1194.78  0.000000
DIFFERENCE     3120.39   2140.48    1058.4    1082.6   1194.78
PROPORTION      0.4117    0.2623    0.1598    0.1091    0.0572    0.0000
CUMULATIVE      0.4117    0.6740    0.8337    0.9428    1.0000    1.0000

               3 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

                              FACTOR PATTERN

                            FACTOR1   FACTOR2   FACTOR3

                   N1      -0.23659   0.44903   0.81477
                   N2       0.73082  -0.08798  -0.45729
                   N3      -0.07492   0.93681  -0.31347
                   N4      -0.71392   0.15703  -0.07350
                   N5      -0.22437  -0.59802  -0.01519
                   N6       0.90824   0.21594   0.21135

                      VARIANCE EXPLAINED BY EACH FACTOR

                             FACTOR1   FACTOR2   FACTOR3
                  WEIGHTED   8596.65   5476.26   3335.78
                  UNWEIGHTED 1.980602  1.515894  1.021533

               FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
    TOTAL COMMUNALITY: WEIGHTED =   17408.7   UNWEIGHTED =   4.518030

                  N1        N2        N3        N4        N5        N6
COMMUNALITY  0.921457  0.750958  0.981484  0.539740  0.408201  0.916190
WEIGHT           3000   2712.02   4925.92   3711.13   422.074   6109.71
```

```
ROTATION METHOD: VARIMAX

                      ORTHOGONAL TRANSFORMATION MATRIX

                                1          2          3

                      1      0.96530    0.08168   -0.24805
                      2     -0.00667    0.95723    0.28925
                      3      0.26107   -0.27755    0.92456

                         ROTATED FACTOR PATTERN

                            FACTOR1    FACTOR2    FACTOR3

                  N1        -0.01866    0.18436    0.94187
                  N2         0.58666    0.10240   -0.62952
                  N3        -0.16041    0.97763   -0.00026
                  N4        -0.70938    0.11240    0.15456
                  N5        -0.21656   -0.58655   -0.13137
                  N6         0.93045    0.22222    0.03258

                  VARIANCE EXPLAINED BY EACH FACTOR

                             FACTOR1    FACTOR2    FACTOR3
                  WEIGHTED    8237.95    5332.18    3838.56
                  UNWEIGHTED 1.786114   1.406289   1.325626

               FINAL COMMUNALITY ESTIMATES AND VARIABLE WEIGHTS
           TOTAL COMMUNALITY: WEIGHTED =   17408.7   UNWEIGHTED =  4.518030

                  N1         N2         N3         N4         N5         N6
COMMUNALITY  0.921457   0.750958   0.981484   0.539740   0.408201   0.916190
WEIGHT           3000    2712.02    4925.92    3711.13    422.074    6109.71
```

Appendix L. Computer Program Code

```
10 '
20 ' THIS PROGRAM DECOMPOSES A DISCRETE EVENT SIMULATION MODEL
30 ' AND AIDS IN ANALYZING THE MODEL'S STRUCTURE
40 '
50 ' THE PROGRAM WAS WRITTEN BY SCOTT MATTHES AT AFIT/EN, WPAFB, OH
60 ' AS PART OF MY MASTER'S THESIS EFFORT
70 '
80 '  INITIALIZATION
90 '
100 WSTR$="##  \                               \       ##
\                             \"
110 WST$ ="NODE         ATTRIBUTE                     NODE
ATTRIBUTE"
120 MATH$="     ATTRIBUTE                   NODE  C/I/O     NODE
C/I/O   W2    W3"
130 MATP$="\                            \ ,  ##    ##         ##    ##
###  #####"
140 BIG = 0
150 '
160 ' OPEN NODE COMPOSITION FILE
170 '
180 OPEN "NODE.CMP" FOR OUTPUT AS #5
190 KEY OFF
200 DIM RSVW$(100), A$(2000)
210 '
220 ' READ IN RESERVE WORDS
230 '
240 RWDN = 0
250 OPEN "RESERVE.WRDS" FOR INPUT AS #1
260 WHILE NOT EOF(1)
270 RWDN = RWDN + 1
280 LINE INPUT#1,RSVW$(RWDN)
290 WEND
300 CLOSE #1
310 '
320 ' ADD RESERVE WORDS
330 '
340 CLS
350 LOCATE 10,20:INPUT "HOW MANY KEY WORDS DO YOU HAVE?   ",KW
360 DIM KWRD$(KW)
370 CLS
380 FOR J=1 TO KW
390 LOCATE 10,20:PRINT "KEY WORD NUMBER";J;:INPUT"IS :   ",KWRD$(J)
400 CLS
410 NEXT J
420 INAT$ = "   INPUT ATTRIBUTE
\                                                \"
430 OTAT$ = "  OUTPUT ATTRIBUTE
\                                                \"
440 K=1
```

```
450 REM NN$ IS NODE NAME
460 CV$ = " CONTROL ATTRIBUTE
\                                                    \"
470 NN$ = " NODE ##
\                                                                \"
480 CLS
490 '
500 ' ENTER FILE TO BE DECOMPOSED
510 '
520 LOCATE 10,20:PRINT "NAME OF FILE TO BE DECOMPOSED:  ":LOCATE
12,26:INPUT " ",INPF$
530 CLS
540 '
550 '   WORK ON NODES
560 '
570 FOR KTR = 1 TO 2
580 NE = 0
590 NODE = 0
600 NL = 0
610 OPEN INPF$ FOR INPUT AS #1
620 WHILE NOT EOF(1)
630 NL = NL + 1
640 LINE INPUT#1,A$(NL)
650 WEND
660 CLOSE #1
670 FOR I=1 TO NL
680 BL2 = 0
690 BL=INSTR(A$(I),"{")
700 EL=INSTR(A$(I),"}")
710 '
720 '   NODE NAMES
730 '
740 IF BL = 0 THEN GOTO 870
750 BN=BL+1
760 EN=EL-BN
770 N$ = MID$(A$(I),BN,EN)
780 IF BL<>0 THEN CN$=N$
790 IF KTR = 2 THEN 830
800 IF NI > BIG THEN BIG = NI
810 IF NO > BIG THEN BIG = NO
820 IF NC > BIG THEN BIG = NC
830 NI = 0
840 NO = 0
850 NC = 0
860 NODE = NODE + 1
870 IF CN$="INITIALIZATION" OR CN$="Initialization" OR CN$="TERMINATION"
OR CN$="Termination" THEN 3160
880 IF BL=0 THEN GOTO 1050
890 IF KTR = 1 THEN LOCATE 10,20 : PRINT "FINDING ATTRIBUTES"
900 IF KTR = 1 THEN 1010
910 IF K=1 THEN 960
920 LOCATE 23,28:PRINT "TYPE ANY KEY TO CONTINUE"
```

```
930 PRINT#5,:PRINT#5,:PRINT#5,
940 WHILE INKEY$="":WEND
950 CLS
960 PRINT USING NN$;NODE,N$
970 PRINT#5,USING NN$;NODE,N$
980 PRINT:PRINT
990 PRINT#5,:PRINT#5,
1000 K=K+1
1010 GOTO 3160
1020 '
1030 ' CREATE STATEMENT
1040 '
1050 BL=INSTR(A$(I),"CREATE")
1060 IF BL=0 THEN 1260
1070 BN=BL+7
1080 MP=INSTR(A$(I),",")
1090 OV$=MID$(A$(I),BN,MP-BN)
1100 NO=NO+1
1110 IF KTR=1 THEN 1220
1120 PRINT USING OTAT$;OV$
1130 EP=INSTR(A$(I),")")
1140 CRSB$=MID$(A$(I),MP+1,EP-MP-1)
1150 OPEN "TEMP.DAT" FOR OUTPUT AS #2
1160 PRINT#2,OV$;"[";CRSB$;"]"
1170 CLOSE #2
1180 OPEN "TEMP.DAT" FOR INPUT AS #3
1190 LINE INPUT#3,WM$(NODE,3,NO)
1200 CLOSE #3
1210 PRINT#5,USING OTAT$;OV$
1220 GOTO 3160
1230 '
1240 ' DESTROY STATEMENT
1250 '
1260 BL=INSTR(A$(I),"DESTROY")
1270 IF BL=0 THEN 1470
1280 BN=BL+8
1290 MP=INSTR(A$(I),",")
1300 IV$=MID$(A$(I),BN,MP-BN)
1310 NI=NI+1
1320 IF KTR=1 THEN 1430
1330 PRINT USING INAT$;IV$
1340 EP=INSTR(A$(I),")")
1350 DESB$=MID$(A$(I),MP+1,EP-MP-1)
1360 OPEN "TEMP.DAT" FOR OUTPUT AS #2
1370 PRINT#2,IV$;"[";DESB$;"]"
1380 CLOSE #2
1390 OPEN "TEMP.DAT" FOR INPUT AS #3
1400 LINE INPUT#3,WM$(NODE,2,NI)
1410 CLOSE #3
1420 PRINT#5,USING INAT$;IV$
1430 GOTO 3160
1440 '
```

```
1450 '   NODE CONTENTS
1460 '
1470 BL=INSTR(A$(I),"WHEN ALARM")
1480 IF BL=0 THEN 1650
1490 KIND = 1
1500 SOP=INSTR(A$(I),"((")
1510 IF SOP=0 THEN 1620
1520 ESP=INSTR(SOP,A$(I),")")
1530 CVW$=MID$(A$(I),SOP+2,ESP-SOP-2)
1540 NC = NC + 1
1550 IF KTR = 1 THEN 1590
1560 PRINT USING CV$;CVW$
1570 WM$(NODE,1,NC) = CVW$
1580 PRINT#5, USING CV$;CVW$
1590 LOOP=0
1600 ECV=ESP-2
1610 GOTO 1880
1620 BN=BL+11
1630 GOTO 1710
1640 GOTO 3160
1650 BL=INSTR(A$(I),"END WHEN")
1660 IF BL<>0 THEN 3160
1670 BL=INSTR(A$(I),"WHEN")
1680 IF BL=0 THEN 2160
1690 KIND = 2
1700 BN=BL+5
1710 EN = INSTR(A$(I),");")
1720 IF EN = 0 THEN EN = INSTR(A$(I),";")
1730 IF EN = 0 THEN GOSUB 7210
1740 SPCH = 1
1750 '
1760 ' TEST FOR OPERATOR
1770 '
1780 LOOP = 0
1790 GOSUB 6340
1800 IF ECV = 0 AND LOOP = 0 THEN 2060
1810 IF ECV = 0 THEN 3160
1820 CVW$ = MID$(A$(I),BN,ECV-BN)
1830 NC = NC + 1
1840 IF KTR = 1 THEN 1880
1850 PRINT USING CV$;CVW$
1860 WM$(NODE,1,NC) = CVW$
1870 PRINT#5,USING CV$;CVW$
1880 SPCH = ECV + 2
1890 BN = INSTR(SPCH,A$(I)," & ")+3
1900 FOUR = 4
1910 IF BN = 3 THEN BN = INSTR(SPCH,A$(I),");")+3
1920 IF BN = 3 THEN BN = INSTR(SPCH,A$(I),";")+3
1930 IF BN = 3 THEN 3160
1940 IF ESP<>0 THEN 2030
1950 IF CASE = 2 THEN BN = BN + 1 : FOUR = FOUR + 1 : SPCH = SPCH + 1
1960 IV$=MID$(A$(I),SPCH+1,BN-SPCH-FOUR)
```

```
1970 GOSUB 6990
1980 IF TEST=0 THEN NI = NI + 1
1990 IF TEST=0 AND KTR = 1 THEN 2030
2000 IF TEST=0 THEN PRINT USING INAT$;IV$
2010 IF TEST=0 THEN WM$(NODE,2,NI) = IV$
2020 IF TEST=0 THEN PRINT#5,USING INAT$;IV$
2030 LOOP = LOOP + 1
2040 ESP=0
2050 GOTO 1790
2060 CVW$ = MID$(A$(I),BN,EN-BN)
2070 NC = NC + 1
2080 IF KTR = 1 THEN 2120
2090 PRINT USING CV$;CVW$
2100 WM$(NODE,1,NC) = CVW$
2110 PRINT#5,USING CV$;CVW$
2120 GOTO 3160
2130 '
2140 '   SET ALARM STATEMENT
2150 '
2160 BL=INSTR(A$(I),"SET ALARM")
2170 IF BL=0 THEN 2600
2180 KIND = 3
2190 BN=BL+10
2200 SOP=INSTR(BN+3,A$(I),"(")
2210 COMA=INSTR(A$(I),", ")
2220 IF SOP > COMA THEN 2250
2230 IF SOP>0 THEN CP=INSTR(BN+3,A$(I),")")
2240 IF SOP>0 THEN MP=INSTR(CP,A$(I),",") : GOTO 2270
2250 MP=INSTR(A$(I),"], ")+1
2260 IF MP < 2 THEN MP=INSTR(A$(I),",")
2270 OV$=MID$(A$(I),BN,MP-BN)
2280 NO = NO + 1
2290 IF KTR = 1 THEN 2330
2300 PRINT USING OTAT$;OV$
2310 WM$(NODE,3,NO) = OV$
2320 PRINT#5,USING OTAT$;OV$
2330 EP=INSTR(A$(I),");")
2340 EPP=INSTR(20,A$(I),"                        ")
2350 IF EP<>0 THEN 2490
2360 MP=INSTR(A$(I),"], ")+1
2370 IF MP < 2 THEN MP=INSTR(A$(I),",")
2380 A$(I)=MID$(A$(I),MP+2,60)
2390 BN=1
2400 GOSUB 7210
2410 GOSUB 6480
2420 NI = NI + 1
2430 IF KTR = 1 THEN 2470
2440 PRINT USING INAT$;IV$
2450 WM$(NODE,2,NI) = IV$
2460 PRINT#5,USING INAT$;IV$
2470 TEMP$=""
2480 GOTO 2590
```

```
2490 IV$=MID$(A$(I),MP+2,EP-MP-2)
2500 GOSUB 6480
2510 GOSUB 6990
2520 IF TEST=1 THEN 3160
2530 IF IV$="" THEN 2590
2540 NI = NI + 1
2550 IF KTR = 1 THEN 2590
2560 PRINT USING INAT$;IV$
2570 WM$(NODE,2,NI) = IV$
2580 PRINT#5,USING INAT$;IV$
2590 GOTO 3160
2600 '
2610 '   INPUT ATTRIBUTES
2620 '
2630 BL = INSTR(A$(I),":=")
2640 IF BL=0 THEN 3160
2650 KIND = 4
2660 HOLD$=A$(I)
2670 BN=BL+4
2680 EN=INSTR(A$(I),";")
2690 IF EN<>0 THEN 2810
2700 GOSUB 7210
2710 GOSUB 6720
2720 EN = INSTR(IV$,")")
2730 IF EN = 0 THEN EI = INSTR(IV$,";")
2740 IV$ = MID$(IV$,BN,EN-BN)
2750 NI = NI + 1
2760 IF KTR = 1 THEN 2800
2770 PRINT USING INAT$;IV$
2780 WM$(NODE,2,NI) = IV$
2790 PRINT#5,USING INAT$;IV$
2800 GOTO 3040
2810 IV$ = MID$(A$(I),BN,EN-BN)
2820 GOSUB 6500
2830 GOSUB 6720
2840 GOSUB 6990
2850 IF TEST = 1 THEN 3040
2860 NI = NI + 1
2870 IF KTR = 1 THEN 2910
2880 PRINT USING INAT$;IV$
2890 WM$(NODE,2,NI) = IV$
2900 PRINT#5,USING INAT$;IV$
2910 IV$=TEMP$
2920 TEMP$=""
2930 GOSUB 6990
2940 IF TEST = 1 THEN 3040
2950 IF IV$="" THEN 3040
2960 NI = NI + 1
2970 IF KTR = 1 THEN 3040
2980 PRINT USING INAT$;IV$
2990 WM$(NODE,2,NI) = IV$
3000 PRINT#5,USING INAT$;IV$
```

```
3010 '
3020 '   OUTPUT ATTRIBUTES
3030 '
3040 FOR J=1 TO 50
3050 IF MID$(HOLD$,J,1)=" " THEN 3070
3060 BO=J:J=50
3070 NEXT J
3080 EO=BL-BO-1
3090 KIND = 5
3100 OV$=MID$(HOLD$,BO,EO)
3110 NO = NO + 1
3120 IF KTR = 1 THEN 3160
3130 PRINT USING OTAT$;OV$
3140 WM$(NODE,3,NO) = OV$
3150 PRINT#5,USING OTAT$;OV$
3160 NEXT I
3170 IF KTR = 1 THEN BIG=BIG+4:DIM
WM$(NODE,3,BIG),WMR$(NODE,3,BIG),SUB$(NODE,3,BIG,7),SUBS$(NODE*3*BIG*7),T
E(NODE,3),SB(NODE,3,BIG)
3180 IF KTR=2 THEN LOCATE 23,28:PRINT "TYPE ANY KEY TO CONTINUE":WHILE
INKEY$="":WEND
3190 CLS
3200 NEXT KTR
3210 CLOSE #5
3220 '
3230 ' FIND SUBSCRIPTS
3240 '
3250 LOCATE 10,20:PRINT"IDENTIFYING SUBCRIPTS OF ATTRIBUTES"
3260 OPEN "EDGE.INF" FOR OUTPUT AS #4
3270 NOV = 0
3280 ERASE A$,RSVW$,KWRD$
3290 FOR A=1 TO NODE
3300 FOR B=1 TO 3
3310 TE(A,B)=0
3320 FOR C=1 TO BIG
3330 SB(A,B,C)=1
3340 NEXT C
3350 NEXT B
3360 NEXT A
3370 DIM NSUB(NODE,3,BIG)
3380 FOR C = 1 TO NODE
3390 FOR B = 1 TO 3
3400 FOR A = 1 TO BIG
3410 IF WM$(C,B,A)="" THEN 3500
3420 PRINT#4, USING "## #  ##  ";C,B,A;:PRINT#4,WM$(C,B,A)
3430 BEG=INSTR(WM$(C,B,A),"[")
3440 IF BEG = 0 THEN WMR$(C,B,A)=WM$(C,B,A):GOTO 3490
3450 WMR$(C,B,A)=MID$(WM$(C,B,A),1,BEG-1)
3460 PRINT#4, USING "## #  ##  ";C,B,A;:PRINT#4,WMR$(C,B,A)
3470 GOSUB 7390
3480 NSUB(C,B,A)=D-FS
3490 NOV = NOV + 1
```

```
3500 NEXT A
3510 NEXT B
3520 NEXT C
3530 CLOSE#4
3540 ERASE TE
3550 E=0
3560 FOR A =1 TO NODE
3570 FOR B =1 TO 3
3580 FOR C =1 TO BIG
3590 FOR D =1 TO 5
3600 IF SUB$(A,B,C,D) = "" THEN 3630
3610 E=E+1
3620 SUBS$(E)=SUB$(A,B,C,D)
3630 NEXT D
3640 NEXT C
3650 NEXT B
3660 NEXT A
3670 DIM SU$(E)
3680 F=0
3690 FOR A=1 TO E
3700 FOR B=1 TO E
3710 IF SUBS$(A)=SUBS$(B) THEN TMP$=SUBS$(A) ELSE 3770
3720 CK = 0
3730 FOR C=1 TO F
3740 IF TMP$=SU$(C) THEN CK=2:C=F
3750 NEXT C
3760 IF CK=0 THEN F=F+1:SU$(F)=TMP$
3770 NEXT B
3780 NEXT A
3790 ERASE SUBS$
3800 SUBS=F
3810 DIM SUBS$(F),SUB(F)
3820 FOR G=1 TO F
3830 SUBS$(G) = SU$(G)
3840 FOR A=0 TO 9
3850 TEST = 0
3860 IF SUBS$(G)<>CHR$(48+A) THEN 3900
3870 SUB(G)=1
3880 TEST=1
3890 A=9
3900 NEXT A
3910 IF TEST=1 THEN 3950
3920 IF G=1 THEN BEEP:BEEP:BEEP
3930 LOCATE 10,10:PRINT"ENTER A MAXIMUM VALUE FOR THE SUBSCRIPT
";SUBS$(G);:INPUT "   ";SUB(G)
3940 CLS
3950 NEXT G
3960 '
3970 ' FIND NETWORK REPRESENTATION
3980 '
3990 LOCATE 10,20:PRINT"FINDING NETWORK REPRESENTATION"
4000 ERASE SU$
```

```
4010 FOR A=1 TO NODE
4020 FOR B=1 TO 3
4030 FOR C=1 TO BIG
4040 FOR D=1 TO 7
4050 FOR G=1 TO F
4060 IF SUB$(A,B,C,D)<>SUBS$(G) THEN 4090
4070 SB(A,B,C)=SB(A,B,C)*SUB(G)
4080 G=F
4090 NEXT G
4100 NEXT D
4110 NEXT C
4120 NEXT B
4130 NEXT A
4140 '
4150 ' WRITE DIAGNOSTICS
4160 '
4170 OPEN "INF.DAT" FOR OUTPUT AS #4
4180 PRINT#4, MATH$
4190 DIM W1(NODE,NODE),CM(NODE,NODE),W2(NODE,NODE)
4200 FOR C=1 TO NODE
4210 FOR D=1 TO NODE
4220 W1(C,D)=0
4230 W2(C,D)=0
4240 NEXT D
4250 NEXT C
4260 OPEN "ERRORS.DAT" FOR OUTPUT AS #5
4270 PRINT#5,"NON-MATCHING SUBSCRIPTS":PRINT#5,
4280 PRINT#5,WST$:PRINT#5,:PRINT#5,
4290 FOR C=1 TO NODE
4300 FOR B=1 TO BIG
4310 FOR A=1 TO BIG
4320 IF WMR$(C,1,B)=WMR$(C,2,A) AND WMR$(C,1,B)<>"" THEN WMR$(C,2,A)=""
4330 IF A <= B THEN 4370
4340 IF WMR$(C,1,B)=WMR$(C,1,A) AND WMR$(C,1,B)<>"" THEN WMR$(C,1,A)=""
4350 IF WMR$(C,2,B)=WMR$(C,2,A) AND WMR$(C,2,B)<>"" THEN WMR$(C,2,A)=""
4360 IF WMR$(C,3,B)=WMR$(C,3,A) AND WMR$(C,3,B)<>"" THEN WMR$(C,3,A)=""
4370 NEXT A
4380 NEXT B
4390 NEXT C
4400 FOR C = 1 TO (NODE-1)
4410 FOR B = 1 TO 3
4420 FOR A = 1 TO BIG
4430 IF WMR$(C,B,A) = "" THEN 4680
4440 FOR X=1 TO NODE
4450 IF X < C THEN 4670
4460 FOR XY = 1 TO 3
4470 IF (X=C AND XY<B) OR (B=XY) OR (B=1 AND XY=2) OR (B=2 AND XY=1)
THEN 4660
4480 FOR Y=1 TO BIG
4490 IF X=C AND XY=B AND Y=C THEN 4650
4500 IF WMR$(C,B,A) <> WMR$(X,XY,Y) OR C = X THEN 4650
```

```
4510 IF WM$(C,B,A)<>WM$(X,XY,Y) AND WM$(C,B,A)<>"" AND WM$(X,XY,Y)<>""
AND NSUB(C,B,A)<>NSUB(X,XY,Y) THEN PRINT#5,"FATAL ERROR!!! Subscripts
are not compatible! Results will be inaccurate!!!"
4520 IF WM$(C,B,A)<>WM$(X,XY,Y) AND WM$(C,B,A)<>"" AND WM$(X,XY,Y)<>""
AND NSUB(C,B,A)<>NSUB(X,XY,Y) THEN PRINT#5,USING
WSTR$;C,WM$(C,B,A),X,WM$(X,XY,Y):PRINT#5,:GOTO 4650
4530 IF WM$(C,B,A) <> WM$(X,XY,Y) AND WM$(C,B,A) <> "" AND WM$(X,XY,Y)
<> "" THEN PRINT#5,"Warning: Subscripts do not match
exactly":PRINT#5,USING WSTR$;C,WM$(C,B,A),X,WM$(X,XY,Y):PRINT#5,
4540 CK1=0
4550 FOR L = 1 TO NSUB(C,B,A)
4560 IF SUB$(C,B,A,L)=SUB$(X,XY,Y,L) THEN 4580
4570 IF SUB$(C,B,A,L)>"/" AND SUB$(C,B,A,L)<":" AND SUB$(X,XY,Y,L)>"/"
AND SUB$(X,XY,Y,L)<":" THEN CK1=1:L=NSUB(C,B,A):PRINT#5,"   *** The
above edge has been disregarded ***":PRINT#5,
4580 NEXT L
4590 IF CK1=1 THEN 4650
4600 W1(C,X) = W1(C,X) + 1
4610 IF B <> 3 THEN SBS = SB(C,B,A) ELSE SBS = SB(X,XY,Y)
4620 W2(C,X) = W2(C,X) + SBS
4630 PRINT#4, USING MATP$;WMR$(C,B,A),C,B,X,XY,W1(C,X),W2(C,X)
4640 EDG = EDG + 1:Y=BIG:XY=3
4650 NEXT Y
4660 NEXT XY
4670 NEXT X
4680 NEXT A
4690 NEXT B
4700 NEXT C
4710 ERASE SUB$
4720 PRINT#4,:PRINT#4,"NUMBER OF EDGES =";EDG
4730 CLOSE #4
4740 NWE=0
4750 FOR C=1 TO NODE
4760 FOR D=1 TO NODE
4770 IF W1(C,D)>0 THEN NWE=NWE+1
4780 NEXT D
4790 NEXT C
4800 CLS
4810 LOCATE 10,20:PRINT"CREATING EDGE MATRIX"
4820 OPEN "INF.DAT" FOR INPUT AS #4
4830 LINE INPUT#4, D$
4840 DIM B$(EDG)
4850 FOR A = 1 TO EDG
4860 INPUT#4, B$(A),D,D,D,D,D,D
4870 NEXT A
4880 CLOSE #4
4890 PRINT#5,CHR$(12):PRINT#5,"ATTRIBUTES WITHOUT A
COUNTERPART":PRINT#5,:PRINT#5,"
ATTRIBUTE                          NODE     TYPE":PRINT#5,:PRINT#5,
4900 FOR D = 1 TO NODE
4910 FOR E = 1 TO 3
4920 FOR F = 1 TO BIG
```

```
4930 MACH = 0
4940 FOR C = 1 TO EDG
4950 IF B$(C) = WMR$(D,E,F) THEN MACH = 1:C=EDG
4960 NEXT C
4970 IF MACH=1 THEN 5030
4980 IF MACH = 0 AND WMR$(D,E,F) = "" THEN 5030
4990 IF E=1 THEN TYP$="CONTROL"
5000 IF E=2 THEN TYP$="INPUT"
5010 IF E=3 THEN TYP$="OUTPUT"
5020 IF MACH = 0 THEN PRINT#5, USING
"\                                          \      ##      \
\";WMR$(D,E,F),D,TYP$
5030 NEXT F
5040 NEXT E
5050 NEXT D
5060 CLOSE #5
5070 ERASE B$,WM$,WMR$
5080 DIM EM(NODE,NWE),WM(NWE,NWE),IM(NODE,NWE)
5090 FOR C=1 TO NODE
5100 FOR D=1 TO NWE
5110 EM(C,D)=0
5120 NEXT D
5130 NEXT C
5140 B=1
5150 FOR C=1 TO NODE
5160 FOR D=1 TO NODE
5170 IF W1(C,D)>0 THEN EM(C,B)=1:EM(D,B)=1:B=B+1
5180 NEXT D
5190 NEXT C
5200 '
5210 ' GET WEIGHT FOR MATRIX
5220 '
5230 DIM SUM(NODE),DIA1(NWE),DIA2(NWE)
5240 FOR C=1 TO NODE
5250 SUM(C)=0
5260 FOR D=1 TO NWE
5270 SUM(C)=SUM(C)+EM(C,D)
5280 IF SUM(C)>0 THEN D=NWE
5290 NEXT D
5300 NEXT C
5310 B=0
5320 H1=0:H2=0
5330 FOR C=1 TO NODE
5340 IF SUM(C)=0 THEN 5430
5350 B=B+1
5360 FOR D=1 TO NWE
5370 EM(B,D)=EM(C,D)
5380 NEXT D
5390 FOR A=1 TO NODE
5400 IF W1(C,A)>0 THEN H1=H1+1:DIA1(H1)=W1(C,A)
5410 IF W2(C,A)>0 THEN H2=H2+1:DIA2(H2)=W2(C,A)
5420 NEXT A
```

```
5430 NEXT C
5440 ERASE SUM
5450 NODE=B
5460 CLS
5470 '
5480 ' CREATE SAS FILE
5490 '
5500 LOCATE  3,15:PRINT"OPTIONS OF MATRIX MULTIPLICATIONS"
5510 LOCATE  6,22:PRINT"1.  E x E'"
5520 LOCATE  8,22:PRINT"2.  E x W2 x E'"
5530 LOCATE 10,22:PRINT"3.  E x W3 x E'"
5540 LOCATE 12,22:PRINT"4.  BOTH  1 & 2"
5550 LOCATE 14,22:PRINT"5.  BOTH  1 & 3"
5560 LOCATE 16,22:PRINT"6.  BOTH  2 & 3"
5570 LOCATE 18,22:PRINT"7.  ALL 3  (1 & 2 & 3)"
5580 BEEP:BEEP:BEEP
5590 LOCATE 21,22:INPUT "ENTER CHOICE  ",CH
5600 IF CH > 7 THEN CLS:SYSTEM
5610 NT=1
5620 CLS
5630 LOCATE 10,10:INPUT"ENTER SAS FILENAME (ie COND.SAS)   ",SASF$
5640 OPEN SASF$ FOR OUTPUT AS #2
5650 PRINT#2,"OPTIONS LINESIZE=80;"
5660 CLS
5670 IF CH <> 1 AND CH <> 4 AND CH <> 5 AND CH <> 7 THEN 5890
5680 CLS
5690 LOCATE 3,10:PRINT "FOR E x E'"
5700 LOCATE 10,15:INPUT"ENTER NUMBER OF FACTORS TO BE RETAINED  ",PV
5710 CLS
5720 PRINT#2,"*;"
5730 PRINT#2,"*   E x E';"
5740 PRINT#2,"*;"
5750 PRINT "         E x E'":PRINT
5760 FOR A=1 TO NODE
5770 FOR B=1 TO NODE
5780 CM(A,B)=0
5790 FOR C=1 TO NWE
5800 CM(A,B) = CM(A,B) + EM(A,C) * EM(B,C)
5810 NEXT C
5820 PRINT USING " ##";CM(A,B);
5830 NEXT B
5840 PRINT
5850 NEXT A
5860 GOSUB 8100
5870 BEEP:BEEP:BEEP
5880 LOCATE 1,50:PRINT"*** HIT A KEY TO CONTINUE ***":WHILE
INKEY$="":WEND
5890 DIM DIA(NWE)
5900 IF CH <> 2 AND CH <> 4 AND CH <> 6 AND CH <> 7 THEN 6040
5910 CLS
5920 LOCATE 3,10:PRINT "FOR E x W2 x E'"
5930 LOCATE 10,15:INPUT"ENTER NUMBER OF FACTORS TO BE RETAINED  ",PV
```

```
5940 CLS
5950 PRINT#2,"*;"
5960 PRINT#2,"*   E x W2 x E';"
5970 PRINT#2,"*;"
5980 FOR C=1 TO NWE
5990 DIA(C)=DIA1(C)
6000 NEXT C
6010 GOSUB 7740
6020 BEEP:BEEP:BEEP
6030 LOCATE 1,50:PRINT"*** HIT A KEY TO CONTINUE ***":WHILE
INKEY$="":WEND
6040 IF CH <> 3 AND CH <> 5 AND CH <> 6 AND CH <> 7 THEN 6280
6050 CLS
6060 LOCATE 3,10:PRINT "FOR E x W3 x E'"
6070 LOCATE 10,15:INPUT"ENTER NUMBER OF FACTORS TO BE RETAINED  ",PV
6080 CLS
6090 PRINT#2,"*;"
6100 PRINT#2,"*   E x W3 x E';"
6110 PRINT#2,"*;"
6120 FOR C=1 TO NWE
6130 DIA(C)=DIA2(C)
6140 NEXT C
6150 GOSUB 7740
6160 FOR A=1 TO SUBS
6170 TEST=0
6180 FOR B=0 TO 9
6190 IF SUBS$(A)<>CHR$(48+B) THEN 6220
6200 B=9
6210 TEST=1
6220 NEXT B
6230 IF TEST=0 THEN PRINT USING "\  \ = ###";SUBS$(A),SUB(A)
6240 IF TEST=0 THEN PRINT#2, USING "* \  \ = ### ;";SUBS$(A),SUB(A)
6250 NEXT A
6260 BEEP:BEEP:BEEP
6270 LOCATE 1,50:PRINT"*** HIT A KEY TO CONTINUE ***":WHILE
INKEY$="":WEND
6280 CLS
6290 CLOSE #2
6300 SYSTEM
6310 '
6320 '   OPERATOR TEST ROUTINE
6330 '
6340 CASE = 1
6350 ECV = INSTR(SPCH,A$(I)," = ")
6360 IF ECV <> 0 THEN RETURN
6370 ECV = INSTR(SPCH,A$(I)," > ")
6380 IF ECV <> 0 THEN RETURN
6390 ECV = INSTR(SPCH,A$(I)," < ")
6400 IF ECV <> 0 THEN RETURN
6410 ECV = INSTR(SPCH,A$(I),"<>")
6420 IF ECV <> 0 THEN CASE = 2:ECV=ECV-1:RETURN
6430 ECV = INSTR(SPCH,A$(I),"<=")
```

```
6440 IF ECV <> 0 THEN CASE = 2:ECV=ECV-1:RETURN
6450 ECV = INSTR(SPCH,A$(I),">=")
6460 IF ECV <> 0 THEN CASE = 2:ECV=ECV-1:RETURN
6470 RETURN
6480 '
6490 ' INPUT PARAMETER ROUTINE
6500 '
6510 SAIP = INSTR(IV$,"(") + 1
6520 SAEP = INSTR(IV$,")")
6530 IF SAEP = 0 THEN RETURN
6540 SAP = SAEP - SAIP
6550 IV$ = MID$(IV$,SAIP,SAP)
6560 INPP = INSTR(IV$,", ")
6570 IF INPP = 0 THEN RETURN
6580 TEM$=IV$
6590 TEMP$ = MID$(TEM$,INPP+2,SAEP-INPP-2)
6600 IV$ = MID$(IV$,1,INPP-1)
6610 IF KIND <> 3 THEN RETURN
6620 GOSUB 6990
6630 IF TEST=1 OR IV$="" THEN 6690
6640 NI=NI+1
6650 IF KTR=1 THEN 6690
6660 PRINT USING INAT$;IV$
6670 WM$(NODE,2,NI)=IV$
6680 PRINT#5,USING INAT$;IV$
6690 ' TEMP$ = MID$(TEM$,INPP+2,SAEP-INPP-2)
6700 IV$ = TEMP$
6710 RETURN
6720 SEP = INSTR(IV$,"+")
6730 IF SEP = 0 THEN SEP = INSTR(IV$,"-")
6740 IF SEP = 0 THEN SEP = INSTR(IV$,"*")
6750 IF SEP = 0 THEN SEP = INSTR(IV$,"/")
6760 IF SEP = 0 THEN RETURN
6770 FOR J=1 TO 50
6780 IF MID$(IV$,J,1)=" " THEN 6800
6790 BN = J :J=50
6800 NEXT J
6810 IF SEP = 0 THEN 6510
6820 IVT$ = MID$(IV$,BN,SEP-2)
6830 IF LEFT$(IVT$,1)="(" THEN IVT$=MID$(IV$,BN+1,SEP-3)
6840 HLD$ = MID$(IV$,SEP+2,50)
6850 IV$=IVT$
6860 GOSUB 6990
6870 IF TEST = 1 THEN 6930
6880 NI = NI + 1
6890 IF KTR = 1 THEN 6930
6900 PRINT USING INAT$;IV$
6910 WM$(NODE,2,NI) = IV$
6920 PRINT#5,USING INAT$;IV$
6930 IV$ = HLD$
6940 GOTO 6720
6950 RETURN
```

```
6960 '
6970 '  CHECK FOR RESERVED , KEY WORDS , AND / OR NUMBERS
6980 '
6990 TEST = 0
7000 FOR M = 1 TO RWDN
7010 IF IV$<>RSVW$(M) THEN 7040
7020 M =  RWDN
7030 TEST = 1
7040 NEXT M
7050 IF TEST = 1 THEN RETURN
7060 FOR M = 1 TO KW
7070 IF IV$ <> KWRD$(M) THEN 7100
7080 M = KW
7090 TEST = 1
7100 NEXT M
7110 IF TEST = 1 THEN RETURN
7120 FOR M = 0 TO 9
7130 IF LEFT$(IV$,1) <> CHR$(48+M) THEN 7160
7140 M = 9
7150 TEST = 1
7160 NEXT M
7170 RETURN
7180 '
7190 ' TWO LINE ROUTINE
7200 '
7210 FOR J =  1 TO 50
7220 IF MID$(A$(I+1),J,1)=" " THEN 7240
7230 BL2=J: J=50
7240 NEXT J
7250 IVP1$ = MID$(A$(I),BN,65)
7260 IVP2$ = MID$(A$(I+1),BL2,50)
7270 OPEN "TEMF.DAT" FOR OUTPUT AS #2
7280 PRINT#2,IVP1$;IVP2$
7290 CLOSE #2
7300 OPEN "TEMP.DAT" FOR INPUT AS #3
7310 LINE INPUT#3,IV$
7320 CLOSE #3
7330 A$(I) = IV$ : BN=1
7340 KILL "TEMP.DAT"
7350 RETURN
7360 '
7370 ' SUBSCRIPT ROUTINE
7380 '
7390 D=0
7400 FS=0
7410 BEG=BEG+1
7420 COMA = INSTR(BEG,WM$(C,B,A),"[")
7430 I = 1000
7440 CM=0
7450 IF COMA > 0 THEN I=COMA
7460 COMA = INSTR(BEG,WM$(C,B,A),",")
7470 IF COMA < I AND COMA > 0 THEN I = COMA
7480 COMA = INSTR(BEG,WM$(C,B,A),"]")
7490 IF COMA < I AND COMA > 0 THEN I = COMA:CM=2
7500 IF I = 1000 THEN RETURN
7510 COMA = I
7520 D=D+1
7530 IF COMA-BEG=0 THEN RETURN
7540 SUB$(C,B,A,D)=MID$(WM$(C,B,A),BEG,COMA-BEG)
7550 IF LEN(SUB$(C,B,A,D)) > 3 THEN GOSUB 7610
7560 IF CM < 2 THEN BEG=COMA+1 ELSE BEG=COMA+2
7570 GOTO 7420
7580 '
7590 ' SUBSCRIPT THAT IS NOT A SUBSCRIPT ROUTINE
7600 '
7610 IF SUB$(C,B,A,D)="" THEN 7700
7620 J=B
7630 IF J<3 THEN J=2
7640 WMR$(C,J,BIG-TE(C,J))=SUB$(C,B,A,D)
7650 PRINT#4, USING "##  #  ##  ";C,J,BIG-
TE(C,J);:PRINT#4,"*";WMR$(C,J,BIG-TE(C,J));"*"
7660 PRINT#4,"THE PRECEDING IS A FORMER SUBSCRIPT"
7670 FS=FS+1
7680 SUB$(C,B,A,D)=""
7690 TE(C,J)=TE(C,J)+1
7700 RETURN
7710 '
7720 ' MATRIX MANIPULATION
7730 '
7740 CLS
7750 LOCATE 10,20:PRINT"CREATING WEIGHT MATRIX"
7760 FOR C=1 TO NWE
7770 FOR D=1 TO NWE
7780 WM(C,D)=0
7790 NEXT D
7800 NEXT C
7810 '
7820 FOR C=1 TO NWE
7830 WM(C,C)=DIA(C)
7840 NEXT C
7850 '
7860 CLS
7870 IF NT=2 OR CH=3 OR CH=5 THEN PRINT "      E x W3 x E'" ELSE PRINT
"      E x W2 x E'"
7880 PRINT
7890 FOR A=1 TO NODE
7900 FOR B=1 TO NWE
7910 IM(A,B)=0
7920 FOR C=1 TO NWE
7930 IM(A,B)=IM(A,B) + EM(A,C) * WM(C,B)
7940 NEXT C
7950 NEXT B
7960 NEXT A
7970 '
```

```
7980 FOR A=1 TO NODE
7990 FOR B=1 TO NODE
8000 CM(A,B)=0
8010 FOR C=1 TO NWE
8020 CM(A,B) = CM(A,B) + IM(A,C) * EM(B,C)
8030 NEXT C
8040 PRINT USING " ####";CM(A,B);
8050 NEXT B
8060 PRINT
8070 NEXT A
8080 PRINT
8090 NT = NT + 1
8100 PRINT#2,"DATA   CONDSPEC;"
8110 PRINT#2,"INPUT ";
8120 FOR A=1 TO (NODE-1)
8130 IF A>9 THEN PRINT#2, USING " N## ";A; ELSE PRINT#2, USING "  N#
";A;
8140 NEXT A
8150 IF NODE>9 THEN PRINT#2, USING " N## ;";NODE ELSE PRINT#2, USING "
N# ;";NODE
8160 PRINT#2,"CARDS;"
8170 FOR A=1 TO NODE
8180 FOR B=1 TO NODE
8190 PRINT#2, USING " ####";CM(A,B);
8200 NEXT B
8210 PRINT#2,
8220 NEXT A
8230 PRINT#2,";"
8240 PRINT#2,"PROC PRINT;"
8250 PRINT#2,USING "PROC FACTOR  COV  NFACTORS = ##   ROTATE=VARIMAX
;";PV
8260 PRINT#2,"   VAR  ";
8270 FOR A=1 TO NODE
8280 IF A>9 THEN PRINT#2, USING " N## ";A; ELSE PRINT#2, USING "  N#
";A;
8290 NEXT A
8300 PRINT#2," ;"
8310 RETURN
```

Bibliography

Balci, Osman. "Requirements for Model Development Environments," Computer and Operations Research, 13: 53-67 (January 1986).

Bauer, K. W., B. Kochar, and J. J. Talavage. "Simulation Model Decomposition by Factor Analysis." 1985 Winter Simulation Conference Proceedings. 185-188. San Francisco, CA, December 1985.

Chvatal, V. Linear Programming. New York: W. H. Freeman and Company, 1983.

de Freitas, V. L. B., A. J. M. G. Rodrigues and J. C. F. M. Neves. "Formal Logic as a Model Specification Language for a Discrete Event Simulation," Proceedings of the 1986 Summer Computer Simulation Conference, pp. 844-849, (July 1986).

Dillon, William R. and Matthew Goldstein. Multivariate Analysis Methods and Applications. New York: John Wiley & Sons, 1984.

Doukidis, G. I. and R. J. Paul. "Experiences in Automating the Formulation of Discrete Event Simulation Models," Proceedings of the European Simulation Conference, pp. 79-90, (February 1985).

Lenz, John E. and Joseph J. Talavage. "A Generalized Simulator for Computerized Manufacturing Systems," Technical Paper. Purdue University, West Layfayette, IN, pp. 3-8, (undated).

Nance, R. E. "The Conical Methodology: A Framework for Simulation Model Development," Proceedings of the Methodology and Validation Conference, pp. 38-43, (April 1987).

Nance, Richard E. and C. Michael Overstreet. Diagnostic Assistance Using Digraph Representations of Discrete Event Simulation Model Specifications. Technical Report SRC-86-001. Systems Research Center and Department of Computer Science, Virginia Polytechnic and State University, Blacksburg, Virginia, 1986.

-----. "Exploring the Forms of Model Diagnosis in a Simulation Support Environment," Proceedins of the 1987 Winter Simulation Conference, pp. 590-595, (December 1987).

Overstreet, C. Michael and Richard E. Nance. "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," Communications of the ACM, 28: 190-201 (February 1985).

Talavage, Joseph J. Personal Correspondence. Purdue University, West Layfayette, IN, 5 August 1986.

Wallace, Jack C. "The Control and Transformation Metric: Toward the Measurement of Simulation Model Complexity," _Proceedins of the 1987 Winter Simulation Conference_, pp. 597-602, (December 1987).

Zeiglar, B. P. "Discrete Event Formalism for Model Based Distributed Simulation," _Simulation Series_, _15_: 3-7 (January 1985).

VITA

Captain Scott R. Matthes, [redacted] he attended Okaloosa Walton Junior College in Niceville, Florida. Double-majoring in mathematics and computer information science, he received a Bachelor of Science degree from Troy State University, Troy, Alabama, in 1984. Upon graduation, he received his commission through Air Force ROTC and was assigned to the Air Force Human Resources Laboratory at Wright-Patterson AFB, Ohio. He served there until entering the School of Engineering, Air Force Institute of Technology, in June, 1987.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/ENS/88D-13	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB, OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

Approved for release in accordance with AFR 190-1, [illegible] 12 Jan 1989

11. TITLE (Include Security Classification)
DISCRETE EVENT SIMULATION MODEL DECOMPOSITION

12. PERSONAL AUTHOR(S)
Scott R. Matthes, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1988 December	125

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	03	
12	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Discrete Event Simulation, Simulation, Discrete Event Simulation, Model Decomposition, Principal Components Analysis

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This research focused on interpreting a discrete event simulation model's condition specification primitives and their associated actions. A network representation was created using these condition action pairs (CAPs) as nodes. The arcs, or edges, of the network represent information being transferred such as specific attributes of the CAPs. This network representation was decomposed into smaller networks, or sub-networks, by taking advantage of the structure of the network. The structure of the network was translated via software interface into an edge-incidence matrix (E-matrix). The E-matrix was then transformed into a pseudo-covariance matrix (C-matrix). The C-matrix was used in the creation of a SAS data set which served as the input necessary to do principal components analysis. Two examples were used todemonstrate this procedure.

Thesis Advisor: Major Kenneth W. Bauer, PhD
Associate Professor of Operational Sciences

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Kenneth W. Bauer, Major, USAF	513-255-3362	AFIT/ENS

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE