Title: Optimal Assignment of Resources in a Software Project

Course:
Year: 1988
Author(s): M. Hoskins, R. Martin and K. Spurgin

Report No: P88002

Abstract: This report explores the effects of different personnel assignments in developing a software product. The L.P. model considers the skills of each engineer, the requirements of each task, and the overall resource limitations on the project. Optimal personnel assignment is made within this environment.

OPTIMAL ASSIGNMENT OF RESOURCES IN A SOFTWARE PROJECT

M. Hoskins, B. Martin, and K. Spurgin

EMP - P8802

Optimal Assignment of Resources in a Software Project

ABSTRACT

This project explores the effects of different personnel assignments on the overall effort in creating a software product. A model is developed which represents the skills of each engineer, the requirements of each task, and overall resource limitations on the project. Within this environment the goal is to determine the optimal assignment of personnel.

Optimal Assignment of Resources in a Software Project

Matt Hoskins
Bob Martin
Kerry Spurgin

EAS 543

1. Introduction

Every engineering project is composed of a collection of tasks. Most projects also involve a team of engineers who must work together to accomplish these tasks. Each engineer possesses a unique mix of skills and experience that determine how successful he will be in performing assigned tasks. One of the problems facing an engineering manager is to assign engineers to tasks in a manner that will put the best person for each task on that task. We will attempt to construct a model for this assignment process which allows the manager to develop a detailed description of the engineer's skills and the task requirements, and to allocate engineers as an instance of the generalized LP assignment problem.

2. Assumptions and Limitations

The primary limitation of this model is its failure to include task dependency in the objective. This limitation is accepted to keep the objective function linear. As a result of this limitation the model will determine the most efficient allocation of personnel in the sense that each assignment puts the best available engineer on that task, but will fail to consider the effect of these assignments on the critical path of the project. It may follow that the "best" allocation of resource does not yield the shortest project duration. Extension of the model to include the project structure is an enticing project but beyond our current capabilities.

The software project being modeled is representative of many projects in today's software development environments. Whether being developed for a PC or a mainframe, most software contains the elements we have chosen. To apply this model to other projects, care should be taken to precisely identify the actual tasks and to analyze the task requirements.

3. Model Description

To develop a detailed description of the project tasks and the engineers, we will use a matrix based approach. We will describe each engineer in terms of his skills. A matrix can be formed with each row representing an engineer and each column representing a particular skill. The elements of this matrix will describe how well each engineer has mastered each skill. This matrix will be called the Skills Matrix. We will discuss each skill later in this section.

We also assume that each task can be described in terms of a combination of certain generic types of functions such as analysis, design, etc. A matrix can also be formed in which each column is an individual task and each row is a particular

function. The elements of this matrix will tell how much of each function is required for each task. We will refer to this matrix as the Task Matrix.

A third matrix is required which describes the relationship between the skills and the functional capabilities. In this matrix the rows refer to the previously defined skills and the columns refer to the previously defined functions. The elements of this matrix describe how much of each skill is utilized in performing each function. Since both the skills and the functions are characteristic of general populations, this matrix should be derived from our general knowledge and experience rather than from the specifics of a particular project.

By computing the product of these three matrices, we form a matrix in which each row refers to an engineer and each column refers to a task. The elements of this matrix describe how well each engineer will do on that particular task. We will refer to this matrix as the Efficiency matrix.

3.1 Skills Matrix

The following skills categories attempt to identify the major skill areas that seem to be required to perform engineering tasks. We have attempted to broaden the definitions to address the entire development cycle, assuming that the engineering team plays an active role in the early stages of product definition. In an environment where the engineers are limited to implementation, the model can be adjusted using the Function matrix.

3.1.1 Written Communication

Written communication is defined as the ability to create written documentation. While this naturally includes basic language skills, it also includes the engineer's ability to translate ideas into a specific and precise description of the project.

3.1.2 Oral Communication

Since most of the initial definition is done in meetings with the customer or a marketing group, the engineer should be able to present ideas clearly in a non-technical manner. The ability to keep the discussion at the proper level of detail and listening skills are also factors in this category.

3.1.3 Creativity

While this skill seems more intangible, it is certainly essential. We use this category to cover a broad definition of creativity, assuming that it would apply to all facets of the project.

3.1.4 Implementation Language Knowledge

This skill covers the engineers mastery of the chosen implementation language. Factors to consider in rating this skill are size and complexity of recent similar projects, exposure to advanced training, and general programming experience.

3.1.5 Operating System Knowledge

Because the operating system will have a major effect on the software design, the degree of prior experience with the chosen OS is important to the design phase as well s the implementation phase. This category includes knowledge of the syntax, functions and idiosyncrasies of OS utilities, as well as system development tools.

3.1.6 Application Knowledge

This category includes all knowledge of the goals of the final project. This is perhaps the most specialized knowledge since it is often different for each project. If the project was to produce a payroll system, this category would include knowledge of the companys benefits, withholding rates and interfaces to other internal databases.

3.1.7 Interpersonal Skills

Because programming is a team effort, each engineer must deal effectively with other team members. The ability to work with others in a competitive and creative environment is a significant factor in teams performance. This category includes traits like attitude, ego, cooperation, and degree of commitment to the team's success.

3.1.8 Self Discipline

This somewhat intangible attribute refers to the ability of the engineer to finish a task without neglecting the details. In code implementation it would affect the expected number of bugs created by the modules he wrote. In other tasks, we would see other effects of his attention to detail or lack of it.

3.1.9 Prior Project Experience

While this is closely related to application knowledge, it actually refers to the engineers understanding of the project's phases, the goals of each task and the relative importance of the different tasks. We assume that people do things better the second time around, so an engineer should be more effective on his second project than on the first one, even if the application has changed.

3.2 Functional Tasks

The following functions are a generalized way to categorize the activities involved in software development.

3.2.1 Requirements Analysis

This activity is basically "deciding what to do". It includes all efforts to work with the customer and other parts of the company to identify what the product must accomplish, what constraints exist on the implementation, how the user interface must work, and what acceptance criteria will be used. It is the process by which the general description of the product is translated and expanded into a precise and complete definition of what will be built.

3.2.2 Architecture Definition

This function is described as architecture because it creates a top level design that meets the needs described in the requirements within any constraints which may exist. In some projects, this function may include the specification and choice of computer hardware and operating systems. In others, many constraints may exist and the function involves finding an approach that will accomplish the objectives within those constraints. For a relatively unconstrained project, this function is primarily creative. For a highly constrainted project, it may require more analysis and deduction.

3.2.3 Detailed Design

Detailed design is the process of translating the higher level design into documentation that can be used to guide and coordinate the coding function. While the methods and form of the documentation may vary with different development methodologies, the basic function of constructing a detailed and specific model of the program is usually viewed as a necessity if the implementation is to be done in a controlled manner.

3.2.4 Coding/Implementation

This function is the normal creation of programs that one usually thinks of as being the primary task of a programmer. It includes some design at the lowest level as well as debugging and testing of small portions of the program.

3.2.5 Testing

Testing is distinguished from debugging in that the goal of testing is to determine that the program works. It usually includes development of a series of tests that can be run in a repeatable fashion to verify the program at different levels of completion.

3.2.6 Integration

This function involves bringing together different pieces of the system and verifying that they work together correctly. Because the pieces are usually done by different people, this function requires a higher degree of cooperation than some other functions. As system problems are uncovered it may be difficult to isolate the problem to a specific module. Since no one person is an expert on the entire system, diagnosis of problems becomes more difficult.

3.2.7 Documentation

The documentation function may require that the engineer write the user's guide and other supporting documents. In other situations the engineer may be required to work with professional writers who develop the documents. This function also includes the development of any internal design and maintenance documents that will be used to support the software.

3.3 Task Description

The following tasks reflect the development model which has been most common during the past 20 years. This model recognizes the complexity level of the software and the communication needs of a large team. Thus it first establishes a written description of the product and then translates the product into code.

3.3.1 Develop Requirements Document

The requirements document defines the goals of the project. The task of producing this document requires that the team document their collective understanding of the customers needs. We assume that this task is performed by the project team. In other projects the requirements may be provided by the customer in a Request For Proposal document.

3.3.2 Define User Interface

While the definition of the interface can be done later in the product development cycle, it is advantageous to distribute proposed interfaces as early as possible. Often the existence of an interface will stimulate the discussion of other requirements. This task requires more creativity and customer knowledge than other definition tasks.

3.3.3 Define Operating System Requirements

Unless the software project includes the creation of an OS, the OS will have a dominant influence on the overall design. Thus it is essential to describe the demands that the software will place on the OS as well as the constraints that the OS will place on the software. Areas like interrupt response time, memory needs, and data storage requirements should be examined in this task.

3.3.4 Create Top Level Design

Within the limitations identified in the previous tasks, the team creates a top level design. This design should identify the major software components in some detail. Through successive expansion of the design, each component should be designed until the lowest level components are identified. The degree to which the components should be described and the level of description should reflect the needs of the software team. The goal of the design is to provide a basis for project control and communication.

3.3.5 Develop Functional Design Document

As a way of formalizing the design process, this document describes how the design components will work together to address the requirements previously specified. It serves as an overview to the design and is useful during the maintenance phase.

3.3.6 Prototype User Interface

Unless the user interface is very simple, developing a prototype of the user interface is needed to confirm that the project is still addressing the user's requirements. Often a customer will not read the requirements document or the functional design, so the prototype becomes the first time that the teams receives confirmation that they are producing what the customer really wants. In addition to stimulating communication about requirements, the prototype often uncovers system problems early, tests assumptions about the OS, and reassures the customer.

3.3.7 Develop Detailed Software Design of User Interface, Application and OS

The detailed design describes each component in a precise manner. It identifys all interfaces such as common data structures and calling sequence. It should provide enough information about the internals of the component that an engineer could modify the component with only the detailed design specification and the source code.

3.3.8 Implement design

This collection of tasks involves the actual translation of the detailed design specifications into working code. In reality different implementation tasks would have properties that might make us assign them to specific programmers. We would normally try to group similar tasks to minimize learning time or assign more complex tasks to more senior engineers. To reduce the complexity of the model we have chosen to treat all implementation tasks as essentially equal. The model could easily be extended by expanding the skills and function dimensions.

3.3.9 Test and Integrate

Test and integration is where the components are put together. The integration plan should call for an orderly and systematic integration which allows functions to be tested with a minimum of special test code. A major part of this phase is the effort to solve system problems as a team. This task may also involve measuring system performance and optimizing some of the system components.

3.3.10 Create System Documentation

While the creation of the user manual may be done by an external group of writers, the team may produce additional documentation of the implementation to aid in future maintenance. These documents would describe the tools used to create the system, problems which were avoided but not solved, potential areas for enhancement and any other information that is likely to be lost is the current members of the team move to other assignments.

3.3.11 System Functional Test

This task is the final test of the system against the requirements as specified in the requirements document. It may involve beta site testing and modifications. In some projects this test may stimulate a request for program modifications beyond the scope of the initial definition, thus starting a new development cycle.

4. Data Collection

The efficiency ratings for engineers in this model are derived from a project manager's subjective ratings of an engineer's skills and the manager's assessment of the functional requirements of the project tasks.

The Function Matrix is developed from expert's knowledge about the software development process. This matrix remains the same regardless of the software project allocation problem being run.

The efficiency rating process captures the interrelationships among the various skills and functions required for a software engineering project. By its multilevel nature and reliance on subjective values, this process lends itself to a "Hierarchical Decision Model" approach. The matrices are illustrated in figure 1.

Using this approach, a project manager's quantitative judgements of engineers' skills are elicited using the constant sum method. The manager is first asked to make pairwise comparisons among the skills which characterize a software engineer. Through a series of matrix operations, normalized values for the skill ratings are calculated. Scale correction is achieved by obtaining the project manager's pairwise comparisons among engineers for their relative proficiencies in a particular skill area. From this input the normalized Skill Matrix is derived. A similar set of steps is used to obtain numerical values for the project manager's evaluation of the functional requirements of the project. These judgements are represented in the Task Matrix.

The Skills Matrix is multiplied with the Function Matrix which has been developed from experts'judgement using the same pairwise comparison method described above. The Function Matrix is also in its "normalized form". The product of these two matrices is normalized and multiplied with the Task Matrix to give the Efficiency Matrix (E).

Normalized results for the matrices are presented in figure 2.

David I. Cleland and Dundar F. Kocaoglu, Engineering Management, (New York: McGraw-Hill, 19XX), pp. 449-458.

5. Modeling Process

The model for engineer/task selection is constructed as a simple linear programming model. This model is formulated using the following verbal description:

- Minimize the total engineering time spent on the project.
- Do not exceed the available work time for each individual engineer.
- Complete the individual task work requirements.

Mathematically, the engineer/task selection model is expressed as below with clarification of model elements following:

Objective Function:

Minimize:

$$\sum_{i=1}^{Neng} \sum_{j=1}^{Ntask} X[i,j]$$

Where: i = The Engineer of interest.
j = The particular task.
X[i,j] = The total number of hours spent by Engineer i on task j.

Subject To:

1. $$\sum_{j=1}^{Ntask} X[i,j] <= ET[i]$$ For all Engineers.

Where: Et = The total available time for Engineer i.

2. $$\sum_{i=1}^{Neng} C[i,j]\, X[i,j] >= TT[j]$$ For each Task.

Where: C[i,j] = The adjustment factor from the efficiency matrix for engineer i on task j.

TT[j] = The total effective time required for each task j.

The modeling process begins with three parameter matrices and two labeling matrices. The parameter matrices are defined as ET[i], TT[j], and C[i,j]. The labeling matrices are the names of the engineers and the tasks to be completed.

The ET[i] matrix is the total time that each engineer has available for task work. This input is a simple determination by the Engineering Manager of total time available for task work for each engineer in his staff.

The TT[j] matrix is the total time that would be required of the nominal individual for completion of the task. This input is a projection judgement made by the

Engineering Manager.

The the efficiency matrix is derived from a Hierarchical model, as described above. A diagram of the process which yields the efficiency matrix is presented in figure 3. This matrix is a relations matrix between two distinct categories. In our example, the categories are engineers and tasks to be completed by those engineers. As a reminder, the basis for the generation of the efficiency matrix was the relative times to complete individual tasks. Therefore, the efficiency matrix is the relational comparison of engineers to tasks to be completed. Using the efficiency matrix we may now gather the required input for a formal formulation of the LP problem. This data is shown in figure 4.

The above input matrices are supplemented with an Engineers name and Task name matrices. These are used for clarification of results but do not enter into the calculations. For the purposes of the engineer/task selection problem the efficiency matrix must be transformed. The transformation normalizes each task column to the task column average. The motivation behind this approach is to relate each engineers efficiency on each task to the nominal time to complete the task, which is defined as TT[j]. The resulting transformed matrix is denoted as C[i,j] in the above mathematical model. Figure 5 shows the C matrix.

The above input matrices are then converted into the Linear Programming model shown in figure 6. The model was then entered into the LINDO problem solving package on the Portland State University computer system. After successful execution the output, attached as figure 7, was obtained.

Because of the extensive matrix manipulation required to formulate the problem, the normal LINDO input mechanism was bypassed. Kerry Spurgin wrote a program to calculate the input matrices and to interface the program output to LINDO. The listing of this program is attached in Appendix A, as an example of how to successfully drive the LINDO program from an external program

6. DISCUSSION OF RESULTS:

The LINDO program sucessfully found an optimal value for the objective function at 7429 man-hours. The total nominal man-hours for the collection of tasks was 7720, so the allocation of engineers has allowed us to improve on a nominal solution by 291 man-hours. If we assume that the project was begun with a budget of 6 engineers for 9.5 months as reflected in the constraints, then we may observe that the slack variables for Sam and Paul are 97 and 1112 respectively. These values suggest that, given the proficiencies represented in the skills matrix, the project could almost be done with five engineers, leaving Paul to work on another project. This allocation is what we would expect when we examine the skill matrix and observe that Paul's skills are weakest in most areas.

In addition to the reduced man-hours, we also note that the recommended assignments make sense and reflect the actual assignments that were made on the project. In those areas where differences exist, they can be easily explained by factors which were not included in the model. For example Steve was assigned to an implementation task based on his strength in application knowledge and prior experience. In reality, his lack of self discipline and oral communication reduced his

effectiveness in this role. In retrospect, this assignment could lead us to reconsider the values assigned in the function matrix.

A second run of the program was made with the allowed man-hours constrained to a little over 8 months for each engineer. This is a typical management approach to attempt to reduce the time to market by requiring a more aggressive deadline. The previous solution indicates that a solution within the total man-hours is possible, so it initially seems reasonable to set the target project duration to this date.

The program found an optimum value at 7492. It is significant that the value of the objective function increased. This makes sense because the increase constraints imply that we run out of our more productive resource sooner and are forced to assign less capable people to the task. In reality this situation is made even worse by the increased communication required to have more than one person doing the task, but communication overhead was not modeled. The assignment of personnel in the more constrained model reflects the necessity to use less productive engineers on certain tasks. It is interesting that constructing a project with relaxed constraints actually results in a more efficient project. This observation seems to run counter to the intuition that most managers bring to the scheduling process. The tendency to view schedule creation as a negotiation between the developers and the customers leads us to ask for the product as soon as possible and then to settle on a compromise. To the degree that the model reflects the actual project, we can see how this is a counterproductive approach.

Because of the construction of this model, the interpretation of data from the sensitivity analysis is not straight forward. A valuable extension to the system would be the ability to hold each of the original input matrices constant two at a time and compute the resultant variability in the third matrix which would create the values given by LINDO in the sensitivity analysis. In particular, by projecting the sensitivity values onto the skill matrix we could see the benefit of improving an engineers skills in a particular area. Given a limited education budget, this information would help us direct our resources to the most beneficial training.

Mapping the sensitivity data onto the task matrix would tell us if a task should be approached differently or perhaps help determine the advantage of different development methodologies given a certain set of engineers.

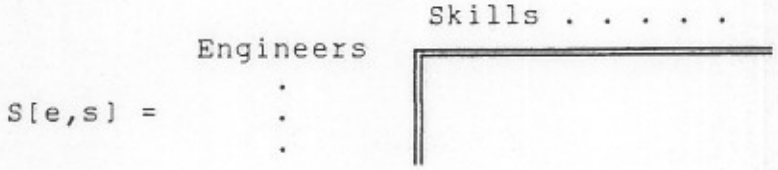
7. Conclusion and Future Work

The initial use of this model seems to establish it as a potential way to approach the allocation of engineers to project tasks. Since most project organization methods assume that all engineers function at the same level, the model could lead to understanding of the dynamics of engineering projects which are currently poorly defined. The extensions to allow project task networks and projection of sensitivity data are the most promising areas for future extensions. Additional areas to incorporate are interpersonal dynamics, communication overhead and learning time on a task basis.

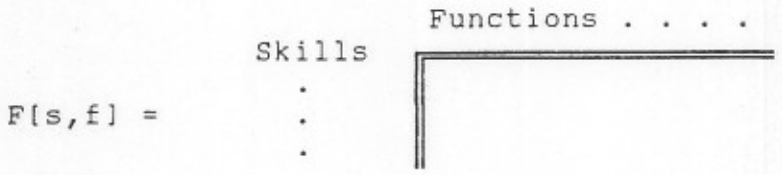
FIGURE 1.

MATRIX FORMAT

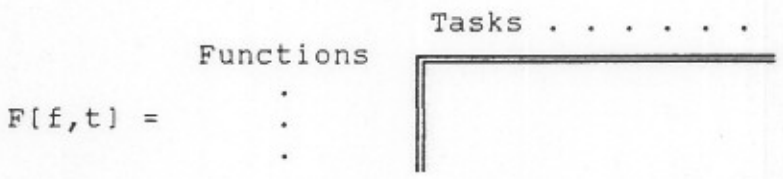
Skills Matrix:



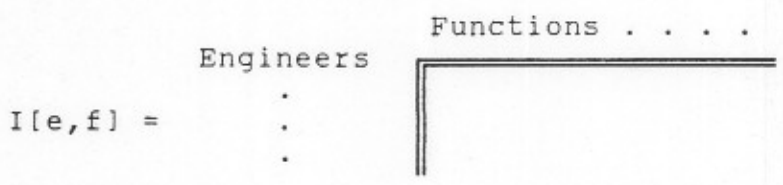
Functions Matrix:



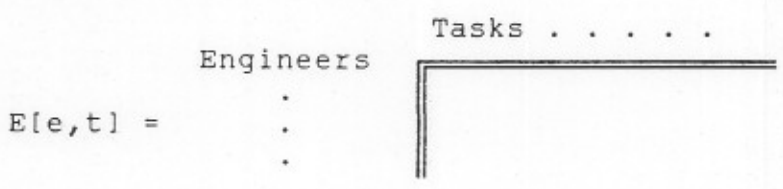
Tasks Matrix:



Intermediate Matrix (Skills X Functions):



Efficiency Matrix (Intermediate X Tasks):



Legend:

S = Skills Matrix (e Engineers by s Skills)
F = Functions Matrix (s Skills by f Functions)
T = Tasks Matrix (f Functions by t Tasks)
I = Intermediate Matrix (e Engineers by f Functions)
E = Efficiency Matrix (e Engineers by t Tasks)

FIGURE 2.

NORMALIZED INPUT MATRICES

SKILLS MATRIX (S), NORMALIZED:

	Skills								
	s1	s2	s3	s4	s5	s6	s7	s8	s9
Engineers									
SAM	2.30	1.15	1.72	2.30	1.72	1.44	0.57	2.30	2.01
STEVE	2.01	1.15	2.30	1.72	1.72	2.59	0.57	0.86	2.59
KEN	2.59	2.59	2.30	2.30	2.30	1.72	2.30	2.59	2.59
JOHN	1.72	2.01	1.44	2.59	2.59	1.72	2.01	0.57	1.44
PAUL	1.44	2.01	2.59	2.30	2.01	1.44	1.72	0.57	1.44
RANDY	2.30	2.30	2.01	2.01	1.72	0.57	2.30	2.30	0.57

FUNCTION MATRIX (F), NORMALIZED:

	Functions						
	f1	f2	f3	f4	f5	f6	f7
Skills							
s1	2.11	1.85	1.85	0.53	0.79	0.79	2.37
s2	1.85	1.58	1.32	0.53	0.79	1.06	2.11
s3	1.58	1.85	2.11	1.32	1.32	0.79	2.37
s4	0.53	1.32	1.85	2.37	2.11	1.85	0.53
s5	0.53	1.85	1.85	2.37	1.85	1.85	0.53
s6	2.11	1.58	1.85	2.37	2.37	2.37	1.32
s7	2.11	1.85	1.06	0.53	0.53	1.58	1.85
s8	1.32	1.32	1.85	2.37	2.37	2.37	1.32
s9	2.37	1.85	1.06	1.06	1.06	1.58	2.37

TASK MATRIX (T), NORMALIZED:

	Tasks										
	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
Functions											
f1	8.11	4.50	2.70	0.90	0.90	0.00	0.00	0.00	0.00	1.80	1.80
f2	0.90	4.50	6.31	7.21	4.50	1.80	1.80	0.00	0.00	0.00	0.00
f3	0.00	0.00	0.00	0.90	4.50	1.80	6.31	1.80	0.00	0.00	0.00
f4	0.00	0.00	0.00	0.00	0.00	5.41	0.90	6.31	1.80	0.00	0.00
f5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	6.31	0.00	0.00
f6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	5.41
f7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.21	1.80

FIGURE 2 (Cont.).

COMPUTED MATRICES FROM INPUT MATRICES

TERMEDIATE MATRIX (S X F), NORMALIZED:

Engineers	Functions f1	f2	f3	f4	f5	f6	f7
SAM	2.17	2.32	2.40	2.26	2.21	2.25	2.25
STEVE	2.34	2.39	2.37	2.19	2.14	2.19	2.38
KEN	3.11	3.22	3.14	2.80	2.76	3.00	3.22
JOHN	2.21	2.45	2.39	2.19	2.10	2.28	2.23
PAUL	2.17	2.38	2.34	2.06	1.99	2.09	2.30
RANDY	2.24	2.42	2.43	2.05	2.03	2.19	2.40

EFFICIENCY MATRIX (E = S x F x T), NORMALIZED:

Engineers	Tasks t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
SAM	1.36	1.39	1.41	1.44	1.60	1.43	1.47	1.42	1.38	1.39	1.39
STEVE	1.46	1.47	1.48	1.48	1.63	1.41	1.47	1.38	1.34	1.47	1.41
KEN	1.94	1.97	1.99	1.99	2.17	1.84	1.94	1.78	1.74	1.99	1.91
JOHN	1.39	1.45	1.48	1.51	1.64	1.42	1.48	1.38	1.33	1.39	1.40
PAUL	1.36	1.42	1.44	1.46	1.60	1.35	1.44	1.31	1.25	1.41	1.34
RANDY	1.41	1.45	1.47	1.49	1.65	1.37	1.49	1.32	1.27	1.47	1.39

NOTE: All table values have been multiplied by 100 for ease in reading.

FIGURE 3.

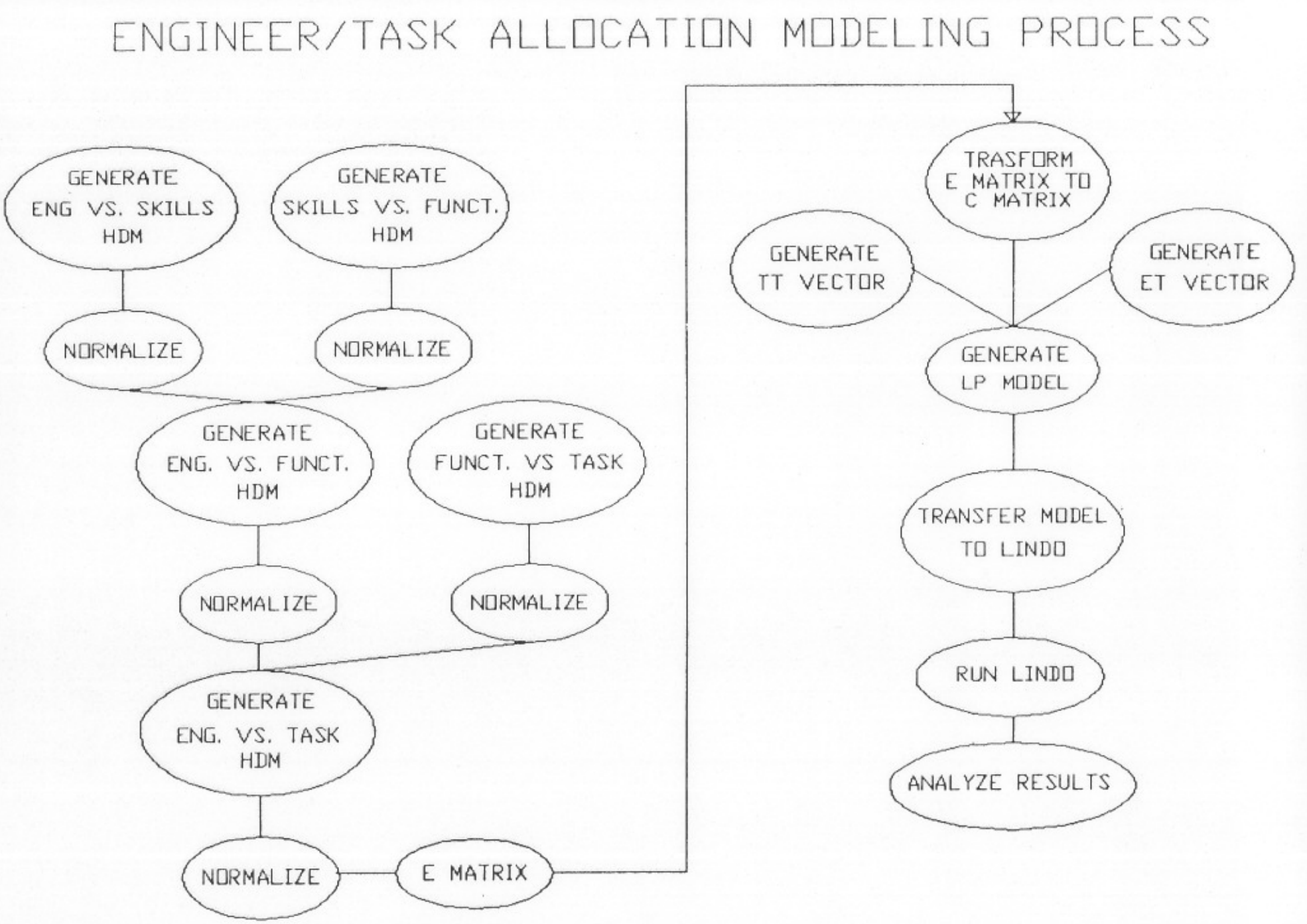

ENGINEER/TASK ALLOCATION MODELING PROCESS
GENERATE ENG VS. SKILLS HDM
GENERATE SKILLS VS. FUNCT. HDM
NORMALIZE
NORMALIZE
GENERATE ENG. VS. FUNCT. HDM
GENERATE FUNCT. VS TASK HDM
NORMALIZE
NORMALIZE
GENERATE ENG. VS. TASK HDM
NORMALIZE
E MATRIX
TRASFORM E MATRIX TO C MATRIX
GENERATE TT VECTOR
GENERATE ET VECTOR
GENERATE LP MODEL
TRANSFER MODEL TO LINDO
RUN LINDO
ANALYZE RESULTS

FIGURE 4.
INPUT TO LPM GENERATION ROUTINE

```
 11                                  (e ENGINEERS, t TASKS)

SAM                                  (ENGINEER NAMES)
STEVE                                (ENGNM[e] MATRIX)
KEN
JOHN
PAUL
RANDY

A                                    (TASK IDENTIFICATION)
B                                    (TASKNM[t] MATRIX)
C
D
E
F
G
H
I
J
K

1.36 1.39 1.41 1.44 1.60 1.43 1.47 1.42 1.38 1.39 1.39   (EFFICIENCY MATRIX)
1.46 1.47 1.48 1.48 1.63 1.41 1.47 1.38 1.34 1.47 1.41   (E[e,t] MATRIX)
1.94 1.97 1.99 1.99 2.17 1.84 1.94 1.78 1.74 1.99 1.91
1 39 1.45 1.48 1.51 1.64 1.42 1.48 1.38 1.33 1.39 1.40
  6 1.42 1.44 1.46 1.60 1.35 1.44 1.31 1.25 1.41 1.34
 41 1.45 1.47 1.49 1.65 1.37 1.49 1.32 1.27 1.47 1.39

 320                     (AVAILABLE ENGINEERING TIME)
 320                     (ET[e] MATRIX)
 320
 400
 300
 200

 900                     (PROJECT TIME FOR TASKS )
 110                     (TT[t] MATRIX)
 130
 700
 120
 800
 950
 750
 130
 130
 800
```

FIGURE 5.

ADJUSTED EFFICIENCY MATRIX
(C[e,t] MATRIX)

	Tasks										
	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
Engineers											
SAM	0.95	0.91	0.90	0.89	0.94	0.95	0.97	0.98	1.01	0.91	0.95
STEVE	0.95	0.98	0.97	0.96	0.94	0.95	0.97	0.98	0.94	0.98	0.95
KEN	1.30	1.30	1.29	1.28	1.29	1.23	1.23	1.26	1.23	1.30	1.30
JOHN	0.95	0.98	0.97	0.96	0.94	0.95	0.97	0.98	0.94	0.91	0.95
PAUL	0.89	0.91	0.90	0.96	0.94	0.95	0.90	0.91	0.94	0.91	0.89
RANDY	0.95	0.91	0.97	0.96	0.94	0.95	0.97	0.91	0.94	0.98	0.95

FIGURE 6.
LINEAR PROGRAMMING MODEL
(LINDO FORMAT)

```
MIN ASAM+BSAM+CSAM
+DSAM+ESAM+FSAM+GSAM
+HSAM+ISAM+JSAM+KSAM+ASTEVE
+BSTEVE+CSTEVE+DSTEVE+ESTEVE
+FSTEVE+GSTEVE+HSTEVE+ISTEVE
+JSTEVE+KSTEVE+AKEN+BKEN+CKEN
+DKEN+EKEN+FKEN+GKEN
+HKEN+IKEN+JKEN+KKEN+AJOHN
+BJOHN+CJOHN+DJOHN+EJOHN
+FJOHN+GJOHN+HJOHN+IJOHN
+JJOHN+KJOHN+APAUL+BPAUL+CPAUL
+DPAUL+EPAUL+FPAUL+GPAUL
+HPAUL+IPAUL+JPAUL+KPAUL+ARANDY
+BRANDY+CRANDY+DRANDY+ERANDY
+FRANDY+GRANDY+HRANDY+IRANDY
+JRANDY+KRANDY
ST
ASAM+BSAM+CSAM
+DSAM+ESAM+FSAM+GSAM
+HSAM+ISAM+JSAM+KSAM <  1440.00
ASTEVE+BSTEVE+CSTEVE
+DSTEVE+ESTEVE+FSTEVE+GSTEVE
+HSTEVE+ISTEVE+JSTEVE+KSTEVE <  1440.00
AKEN+BKEN+CKEN
+DKEN+EKEN+FKEN+GKEN
+HKEN+IKEN+JKEN+KKEN <  1440.00
AJOHN+BJOHN+CJOHN
+DJOHN+EJOHN+FJOHN+GJOHN
+HJOHN+IJOHN+JJOHN+KJOHN <  1440.00
APAUL+BPAUL+CPAUL
+DPAUL+EPAUL+FPAUL+GPAUL
+HPAUL+IPAUL+JPAUL+KPAUL <  1440.00
ARANDY+BRANDY+CRANDY
+DRANDY+ERANDY+FRANDY+GRANDY
+HRANDY+IRANDY+JRANDY+KRANDY <  1440.00
```

FIGURE 6 (Cont.).
LINEAR PROGRAMMING MODEL
(LINDO FORMAT)

```
 0.93ASAM+ 1.00ASTEVE+ 1.27AKEN
+ 0.93AJOHN+ 0.93APAUL+ 0.93ARANDY >    320.00
 0.91BSAM+ 0.98BSTEVE+ 1.30BKEN
+ 0.98BJOHN+ 0.91BPAUL+ 0.91BRANDY >    480.00
 0.90CSAM+ 0.97CSTEVE+ 1.29CKEN
+ 0.97CJOHN+ 0.90CPAUL+ 0.97CRANDY >    160.00
 0.89DSAM+ 0.96DSTEVE+ 1.28DKEN
+ 0.96DJOHN+ 0.96DPAUL+ 0.96DRANDY >    160.00
 0.94ESAM+ 0.94ESTEVE+ 1.29EKEN
+ 0.94EJOHN+ 0.94EPAUL+ 0.94ERANDY >    960.00
 0.95FSAM+ 0.95FSTEVE+ 1.23FKEN
+ 0.95FJOHN+ 0.95FPAUL+ 0.95FRANDY >    320.00
 0.97GSAM+ 0.97GSTEVE+ 1.23GKEN
+ 0.97GJOHN+ 0.90GPAUL+ 0.97GRANDY >   1600.00
 0.98HSAM+ 0.98HSTEVE+ 1.26HKEN
+ 0.98HJOHN+ 0.91HPAUL+ 0.91HRANDY >   2400.00
 1.01ISAM+ 0.94ISTEVE+ 1.23IKEN
+ 0.94IJOHN+ 0.94IPAUL+ 0.94IRANDY >    320.00
 0.91JSAM+ 0.98JSTEVE+ 1.30JKEN
+ 0.91JJOHN+ 0.91JPAUL+ 0.98JRANDY >    800.00
 0.95KSAM+ 0.95KSTEVE+ 1.30KKEN
+ 0.95KJOHN+ 0.89KPAUL+ 0.95KRANDY >    200.00
END
LEAVE
```

FIGURE 7.

LINDO OUTPUT

```
 IN      ASAM + BSAM + CSAM + DSAM + ESAM + FSAM + GSAM + HSAM
       + ISAM + JSAM + KSAM + ASTEVE + BSTEVE + CSTEVE + DSTEVE
       + ESTEVE + FSTEVE + GSTEVE + HSTEVE + ISTEVE + JSTEVE
       + KSTEVE + AKEN + BKEN + CKEN + DKEN + EKEN + FKEN + GKEN
       + HKEN + IKEN + JKEN + KKEN + AJOHN + BJOHN + CJOHN + DJOHN
       + EJOHN + FJOHN + GJOHN + HJOHN + IJOHN + JJOHN + KJOHN
       + APAUL + BPAUL + CPAUL + DPAUL + EPAUL + FPAUL + GPAUL
       + HPAUL + IPAUL + JPAUL + KPAUL + ARANDY + BRANDY + CRANDY
       + DRANDY + ERANDY + FRANDY + GRANDY + HRANDY + IRANDY
       + JRANDY + KRANDY
SUBJECT TO
        2)    ASAM + BSAM + CSAM + DSAM + ESAM + FSAM + GSAM + HSAM
       + ISAM + JSAM + KSAM <=    1440
        3)    ASTEVE + BSTEVE + CSTEVE + DSTEVE + ESTEVE + FSTEVE
       + GSTEVE + HSTEVE + ISTEVE + JSTEVE + KSTEVE <=    1440
        4)    AKEN + BKEN + CKEN + DKEN + EKEN + FKEN + GKEN + HKEN
       + IKEN + JKEN + KKEN <=    1440
        5)    AJOHN + BJOHN + CJOHN + DJOHN + EJOHN + FJOHN + GJOHN
       + HJOHN + IJOHN + JJOHN + KJOHN <=    1440
        6)    APAUL + BPAUL + CPAUL + DPAUL + EPAUL + FPAUL + GPAUL
       + HPAUL + IPAUL + JPAUL + KPAUL <=    1440
        7)    ARANDY + BRANDY + CRANDY + DRANDY + ERANDY + FRANDY
       + GRANDY + HRANDY + IRANDY + JRANDY + KRANDY <=    1440
        8)    0.93 ASAM + ASTEVE + 1.27 AKEN + 0.93 AJOHN
       + 0.93 APAUL + 0.93 ARANDY >=    320
        9)    0.91 BSAM + 0.98 BSTEVE + 1.3 BKEN + 0.98 BJOHN
       + 0.91 BPAUL + 0.91 BRANDY >=    480
       10)    0.9 CSAM + 0.97 CSTEVE + 1.29 CKEN + 0.97 CJOHN
       + 0.9 CPAUL + 0.97 CRANDY >=    160
       11)    0.89 DSAM + 0.96 DSTEVE + 1.28 DKEN + 0.96 DJOHN
       + 0.96 DPAUL + 0.96 DRANDY >=    160
       12)    0.94 ESAM + 0.94 ESTEVE + 1.29 EKEN + 0.94 EJOHN
       + 0.94 EPAUL + 0.94 ERANDY >=    960
       13)    0.95 FSAM + 0.95 FSTEVE + 1.23 FKEN + 0.95 FJOHN
       + 0.95 FPAUL + 0.95 FRANDY >=    320
       14)    0.97 GSAM + 0.97 GSTEVE + 1.23 GKEN + 0.97 GJOHN
       + 0.9 GPAUL + 0.97 GRANDY >=    1600
       15)    0.98 HSAM + 0.98 HSTEVE + 1.26 HKEN + 0.98 HJOHN
       + 0.91 HPAUL + 0.91 HRANDY >=    2400
       16)    1.01 ISAM + 0.94 ISTEVE + 1.23 IKEN + 0.94 IJOHN
       + 0.94 IPAUL + 0.94 IRANDY >=    320
       17)    0.91 JSAM + 0.98 JSTEVE + 1.3 JKEN + 0.91 JJOHN
       + 0.91 JPAUL + 0.98 JRANDY >=    800
       18)    0.95 KSAM + 0.95 KSTEVE + 1.3 KKEN + 0.95 KJOHN
       + 0.89 KPAUL + 0.95 KRANDY >=    200
END
```

FIGURE 7 (Cont.).
LINDO OUTPUT

OPTIMUM FOUND AT STEP 22

OBJECTIVE FUNCTION VALUE

1) 7429.66797

VARIABLE	VALUE	REDUCED COST
ASAM	0.000000	0.070000
BSAM	0.000000	0.071430
CSAM	0.000000	0.074516
DSAM	0.000000	0.077649
ESAM	0.000000	0.033383
FSAM	0.000000	0.000000
GSAM	1025.811040	0.000000
HSAM	0.000000	0.000001
ISAM	316.831787	0.000000
JSAM	0.000000	0.071429
KSAM	0.000000	0.030615
ASTEVE	320.000000	0.000000
BSTEVE	0.000000	0.000001
CSTEVE	0.000000	0.002534
DSTEVE	0.000000	0.005104
ESTEVE	0.000000	0.033383
FSTEVE	9.813589	0.000000
GSTEVE	0.000000	0.000000
HSTEVE	1110.186280	0.000000
ISTEVE	0.000000	0.069306
JSTEVE	0.000000	0.000000
KSTEVE	0.000000	0.030614
AKEN	0.000000	0.056529
BKEN	292.936523	0.000000
CKEN	124.031006	0.000000
DKEN	125.000015	0.000000
EKEN	744.186035	0.000000
FKEN	0.000000	0.031792
GKEN	0.000000	0.058488
HKEN	0.000000	0.040815
IKEN	0.000000	0.108706
JKEN	0.000000	-0.000001
KKEN	153.846237	0.000000
AJOHN	0.000000	0.070000
BJOHN	101.206711	0.000000
CJOHN	0.000000	0.002533
DJOHN	0.000000	0.005104
EJOHN	0.000000	0.033383
FJOHN	0.000000	0.000000
GJOHN	0.000000	0.000000
HJOHN	1338.793210	0.000000
IJOHN	0.000000	0.069306
JJOHN	0.000000	0.071429
KJOHN	0.000000	0.030614
APAUL	0.000000	0.070000
BPAUL	0.000000	0.071430
CPAUL	0.000000	0.074516
DPAUL	0.000000	0.005105
EPAUL	0.000000	0.033383
FPAUL	327.028320	0.000000

FIGURE 7 (Cont.).
LINDO OUTPUT

GPAUL	0.000000	0.072165
HPAUL	0.000000	0.071429
IPAUL	0.000000	0.069306
JPAUL	0.000000	0.071429
KPAUL	0.000000	0.091839
ARANDY	0.000000	0.070000
BRANDY	0.000000	0.071430
CRANDY	0.000000	0.002534
DRANDY	0.000000	0.005104
ERANDY	0.000000	0.033383
FRANDY	0.000000	0.000000
GRANDY	623.673340	0.000000
HRANDY	0.000000	0.071429
IRANDY	0.000000	0.069306
JRANDY	816.326416	0.000000
KRANDY	0.000000	0.030614

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	97.356934	0.000000
3)	0.000000	0.000000
4)	0.000000	0.326528
5)	0.000000	-0.000001
6)	1112.971440	0.000000
7)	0.000000	0.000000
8)	0.000000	-1.000000
9)	0.000000	-1.020407
10)	0.000000	-1.028316
11)	0.000000	-1.036349
12)	0.000000	-1.028316
13)	0.000000	-1.052631
14)	0.000000	-1.030928
15)	0.000000	-1.020408
16)	0.000000	-0.990100
17)	0.000000	-1.020408
18)	0.000000	-1.020406

NO. ITERATIONS= 22

RANGES IN WHICH THE BASIS IS UNCHANGED:

	OBJ COEFFICIENT RANGES		
VARIABLE	CURRENT COEF	ALLOWABLE INCREASE	ALLOWABLE DECREASE
ASAM	1.000000	INFINITY	0.070000
BSAM	1.000000	INFINITY	0.071430
CSAM	1.000000	INFINITY	0.074516
DSAM	1.000000	INFINITY	0.077649
ESAM	1.000000	INFINITY	0.033383
FSAM	1.000000	INFINITY	0.000000
GSAM	1.000000	-0.000001	0.000000
HSAM	1.000000	INFINITY	0.000001
ISAM	1.000000	0.074467	1.000000
JSAM	1.000000	INFINITY	0.071429
KSAM	1.000000	INFINITY	0.030615
ASTEVE	1.000000	0.044511	1.000000

FIGURE 7 (Cont.).
LINDO OUTPUT

BSTEVE	1.000000	INFINITY	0.000001
CSTEVE	1.000000	INFINITY	0.002534
DSTEVE	1.000000	INFINITY	0.005104
ESTEVE	1.000000	INFINITY	0.033383
FSTEVE	1.000000	-0.000001	0.000001
GSTEVE	1.000000	INFINITY	0.000000
HSTEVE	1.000000	0.000001	-0.000001
ISTEVE	1.000000	INFINITY	0.069306
JSTEVE	1.000000	INFINITY	0.000000
KSTEVE	1.000000	INFINITY	0.030614
AKEN	1.000000	INFINITY	0.056529
BKEN	1.000000	-0.000001	0.003369
CKEN	1.000000	0.003369	1.326527
DKEN	1.000000	0.006806	1.326527
EKEN	1.000000	0.045813	1.326527
FKEN	1.000000	INFINITY	0.031792
GKEN	1.000000	INFINITY	0.058488
HKEN	1.000000	INFINITY	0.040815
IKEN	1.000000	INFINITY	0.108706
JKEN	1.000000	INFINITY	-0.000001
KKEN	1.000000	0.041893	1.326527
AJOHN	1.000000	INFINITY	0.070000
BJOHN	1.000000	0.000001	-0.000001
CJOHN	1.000000	INFINITY	0.002533
DJOHN	1.000000	INFINITY	0.005104
EJOHN	1.000000	INFINITY	0.033383
FJOHN	1.000000	INFINITY	0.000000
GJOHN	1.000000	INFINITY	0.000000
HJOHN	1.000000	-0.000001	0.000001
IJOHN	1.000000	INFINITY	0.069306
JJOHN	1.000000	INFINITY	0.071429
KJOHN	1.000000	INFINITY	0.030614
APAUL	1.000000	INFINITY	0.070000
BPAUL	1.000000	INFINITY	0.071430
CPAUL	1.000000	INFINITY	0.074516
DPAUL	1.000000	INFINITY	0.005105
EPAUL	1.000000	INFINITY	0.033383
FPAUL	1.000000	0.000000	-0.000001
GPAUL	1.000000	INFINITY	0.072165
HPAUL	1.000000	INFINITY	0.071429
IPAUL	1.000000	INFINITY	0.069306
JPAUL	1.000000	INFINITY	0.071429
KPAUL	1.000000	INFINITY	0.091839
ARANDY	1.000000	INFINITY	0.070000
BRANDY	1.000000	INFINITY	0.071430
CRANDY	1.000000	INFINITY	0.002534
DRANDY	1.000000	INFINITY	0.005104
ERANDY	1.000000	INFINITY	0.033383
FRANDY	1.000000	INFINITY	0.000000
GRANDY	1.000000	0.000000	-0.000001
HRANDY	1.000000	INFINITY	0.071429
IRANDY	1.000000	INFINITY	0.069306
JRANDY	1.000000	-0.000001	1.000000
KRANDY	1.000000	INFINITY	0.030614

FIGURE 7 (Cont.).
LINDO OUTPUT

RIGHTHAND SIDE RANGES

ROW	CURRENT RHS	ALLOWABLE INCREASE	ALLOWABLE DECREASE
2	1440.000000	INFINITY	97.356934
3	1440.000000	327.028320	9.813589
4	1440.000000	76.294342	7.397943
5	1440.000000	327.028320	9.813589
6	1440.000000	INFINITY	1112.971440
7	1440.000000	1025.811040	97.356934
8	320.000000	9.813589	320.000000
9	480.000000	9.617321	99.182617
10	160.000000	9.543345	98.419708
11	160.000000	9.469366	97.656769
12	960.000000	9.543345	98.419708
13	320.000000	1057.323000	310.676758
14	1600.000000	94.436234	995.036865
15	2400.000000	9.617321	320.487793
16	320.000000	98.330429	319.999756
17	800.000000	95.409836	800.000244
18	200.000000	9.617321	99.182617

THE TABLEAU

ROW	(BASIS)	ASAM	BSAM	CSAM	DSAM
1	ART	0.070	0.071	0.075	0.078
2	SLK 2	1.000	1.000	1.000	1.000
3	FSTEVE	-0.930	-0.929	-0.925	-0.922
4	BKEN	0.000	0.000	-0.698	-0.695
5	HJOHN	0.000	-0.929	-0.925	-0.922
6	SLK 6	-0.930	-0.929	-0.925	-0.922
7	GRANDY	0.000	0.000	0.000	0.000
8	ASTEVE	0.930	0.000	0.000	0.000
9	BJOHN	0.000	0.929	0.925	0.922
10	CKEN	0.000	0.000	0.698	0.000
11	DKEN	0.000	0.000	0.000	0.695
12	EKEN	0.000	0.000	0.000	0.000
13	FPAUL	0.930	0.929	0.925	0.922
14	GSAM	0.000	0.000	0.000	0.000
15	HSTEVE	0.000	0.929	0.925	0.922
16	ISAM	0.000	0.000	0.000	0.000
17	JRANDY	0.000	0.000	0.000	0.000
18	KKEN	0.000	0.000	0.000	0.000

ROW	ESAM	FSAM	GSAM	HSAM	ISAM
1	0.033	0.000	0.000	0.000	0.000
2	1.000	1.000	0.000	1.000	0.000
3	-0.967	0.000	0.000	-1.000	0.000
4	-0.729	0.000	0.000	0.000	0.000
5	-0.967	0.000	0.000	0.000	0.000
6	-0.967	-1.000	0.000	-1.000	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.967	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000

FIGURE 7 (Cont.).
LINDO OUTPUT

11	0.000	0.000	0.000	0.000	0.000
12	0.729	0.000	0.000	0.000	0.000
13	0.967	1.000	0.000	1.000	0.000
14	0.000	0.000	1.000	0.000	0.000
15	0.967	0.000	0.000	1.000	0.000
16	0.000	0.000	0.000	0.000	1.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	JSAM	KSAM	ASTEVE	BSTEVE	CSTEVE
1	0.071	0.031	0.000	0.000	0.003
2	0.071	1.000	0.000	0.000	0.000
3	0.000	-0.969	0.000	0.000	0.003
4	0.000	-0.731	0.000	0.000	-0.752
5	0.000	-0.969	0.000	-1.000	-0.997
6	0.000	-0.969	0.000	0.000	0.003
7	-0.929	0.000	0.000	0.000	0.000
8	0.000	0.000	1.000	0.000	0.000
9	0.000	0.969	0.000	1.000	0.997
10	0.000	0.000	0.000	0.000	0.752
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	0.000	0.969	0.000	0.000	-0.003
14	0.929	0.000	0.000	0.000	0.000
15	0.000	0.969	0.000	1.000	0.997
16	0.000	0.000	0.000	0.000	0.000
17	0.929	0.000	0.000	0.000	0.000
18	0.000	0.731	0.000	0.000	0.000

ROW	DSTEVE	ESTEVE	FSTEVE	GSTEVE	HSTEVE
1	0.005	0.033	0.000	0.000	0.000
2	0.000	0.000	0.000	-1.000	0.000
3	0.005	0.033	1.000	1.000	0.000
4	-0.750	-0.729	0.000	0.000	0.000
5	-0.995	-0.967	0.000	0.000	0.000
6	0.005	0.033	0.000	1.000	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.995	0.967	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000
11	0.750	0.000	0.000	0.000	0.000
12	0.000	0.729	0.000	0.000	0.000
13	-0.005	-0.033	0.000	-1.000	0.000
14	0.000	0.000	0.000	1.000	0.000
15	0.995	0.967	0.000	0.000	1.000
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

FIGURE 7 (Cont.).
LINDO OUTPUT

ROW	ISTEVE	JSTEVE	KSTEVE	AKEN	BKEN
1	0.069	0.000	0.031	0.057	0.000
2	-0.931	-1.000	0.000	0.000	0.000
3	1.000	1.000	0.031	0.057	0.000
4	0.000	0.000	-0.731	1.000	1.000
5	0.000	0.000	-0.969	1.327	0.000
6	1.000	1.000	0.031	0.057	0.000
7	0.000	-1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	1.270	0.000
9	0.000	0.000	0.969	-1.327	0.000
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	-1.000	-1.000	-0.031	-0.057	0.000
14	0.000	1.000	0.000	0.000	0.000
15	0.000	0.000	0.969	-1.327	0.000
16	0.931	0.000	0.000	0.000	0.000
17	0.000	1.000	0.000	0.000	0.000
18	0.000	0.000	0.731	0.000	0.000

ROW	CKEN	DKEN	EKEN	FKEN	GKEN
1	0.000	0.000	0.000	0.032	0.058
2	0.000	0.000	0.000	0.000	-1.268
3	0.000	0.000	0.000	1.327	1.327
4	0.000	0.000	0.000	1.000	1.000
5	0.000	0.000	0.000	1.327	1.327
6	0.000	0.000	0.000	0.032	1.327
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	-1.327	-1.327
10	1.000	0.000	0.000	0.000	0.000
11	0.000	1.000	0.000	0.000	0.000
12	0.000	0.000	1.000	0.000	0.000
13	0.000	0.000	0.000	-0.032	-1.327
14	0.000	0.000	0.000	0.000	1.268
15	0.000	0.000	0.000	-1.327	-1.327
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	HKEN	IKEN	JKEN	KKEN	AJOHN
1	0.041	0.109	0.000	0.000	0.070
2	0.000	-1.218	-1.327	0.000	0.000
3	0.041	1.327	1.327	0.000	0.070
4	1.000	1.000	1.000	0.000	0.000
5	1.327	1.327	1.327	0.000	1.000
6	0.041	1.327	1.327	0.000	0.070
7	0.000	0.000	-1.327	0.000	0.000
8	0.000	0.000	0.000	0.000	0.930
9	-1.327	-1.327	-1.327	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000

FIGURE 7 (Cont.).
LINDO OUTPUT

13	-0.041	-1.327	-1.327	0.000	-0.070
14	0.000	0.000	1.327	0.000	0.000
15	-0.041	-1.327	-1.327	0.000	-1.000
16	0.000	1.218	0.000	0.000	0.000
17	0.000	0.000	1.327	0.000	0.000
18	0.000	0.000	0.000	1.000	0.000

ROW	BJOHN	CJOHN	DJOHN	EJOHN	FJOHN
1	0.000	0.003	0.005	0.033	0.000
2	0.000	0.000	0.000	0.000	0.000
3	0.000	0.003	0.005	0.033	1.000
4	0.000	-0.752	-0.750	-0.729	0.000
5	0.000	0.003	0.005	0.033	1.000
6	0.000	0.003	0.005	0.033	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	1.000	0.997	0.995	0.967	0.000
10	0.000	0.752	0.000	0.000	0.000
11	0.000	0.000	0.750	0.000	0.000
12	0.000	0.000	0.000	0.729	0.000
13	0.000	-0.003	-0.005	-0.033	0.000
14	0.000	0.000	0.000	0.000	0.000
15	0.000	-0.003	-0.005	-0.033	-1.000
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	GJOHN	HJOHN	IJOHN	JJOHN	KJOHN
1	0.000	0.000	0.069	0.071	0.031
2	-1.000	0.000	-0.931	-0.929	0.000
3	1.000	0.000	1.000	1.000	0.031
4	0.000	0.000	0.000	0.000	-0.731
5	1.000	1.000	1.000	1.000	0.031
6	1.000	0.000	1.000	1.000	0.031
7	0.000	0.000	0.000	-0.929	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.969
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	-1.000	0.000	-1.000	-1.000	-0.031
14	1.000	0.000	0.000	0.929	0.000
15	-1.000	0.000	-1.000	-1.000	-0.031
16	0.000	0.000	0.931	0.000	0.000
17	0.000	0.000	0.000	0.929	0.000
18	0.000	0.000	0.000	0.000	0.731

ROW	APAUL	BPAUL	CPAUL	DPAUL	EPAUL
1	0.070	0.071	0.075	0.005	0.033
2	0.000	0.000	0.000	0.000	0.000
3	-0.930	-0.929	-0.925	-0.995	-0.967
4	0.000	0.000	-0.698	-0.750	-0.729

FIGURE 7 (Cont.).
LINDO OUTPUT

5	0.000	-0.929	-0.925	-0.995	-0.967
6	0.070	0.071	0.075	0.005	0.033
7	0.000	0.000	0.000	0.000	0.000
8	0.930	0.000	0.000	0.000	0.000
9	0.000	0.929	0.925	0.995	0.967
10	0.000	0.000	0.698	0.000	0.000
11	0.000	0.000	0.000	0.750	0.000
12	0.000	0.000	0.000	0.000	0.729
13	0.930	0.929	0.925	0.995	0.967
14	0.000	0.000	0.000	0.000	0.000
15	0.000	0.929	0.925	0.995	0.967
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	FPAUL	GPAUL	HPAUL	IPAUL	JPAUL
1	0.000	0.072	0.071	0.069	0.071
2	0.000	-0.928	0.000	-0.931	-0.929
3	0.000	0.000	-0.929	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000
6	0.000	1.000	0.071	1.000	1.000
7	0.000	0.000	0.000	0.000	-0.929
8	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	1.000	0.000	0.929	0.000	0.000
14	0.000	0.928	0.000	0.000	0.929
15	0.000	0.000	0.929	0.000	0.000
16	0.000	0.000	0.000	0.931	0.000
17	0.000	0.000	0.000	0.000	0.929
18	0.000	0.000	0.000	0.000	0.000

ROW	KPAUL	ARANDY	BRANDY	CRANDY	DRANDY
1	0.092	0.070	0.071	0.003	0.005
2	0.000	1.000	1.000	1.000	1.000
3	-0.908	-0.930	-0.929	-0.997	-0.995
4	-0.685	0.000	0.000	-0.752	-0.750
5	-0.908	0.000	-0.929	-0.997	-0.995
6	0.092	-0.930	-0.929	-0.997	-0.995
7	0.000	1.000	1.000	1.000	1.000
8	0.000	0.930	0.000	0.000	0.000
9	0.908	0.000	0.929	0.997	0.995
10	0.000	0.000	0.000	0.752	0.000
11	0.000	0.000	0.000	0.000	0.750
12	0.000	0.000	0.000	0.000	0.000
13	0.908	0.930	0.929	0.997	0.995
14	0.000	-1.000	-1.000	-1.000	-1.000
15	0.908	0.000	0.929	0.997	0.995
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000

FIGURE 7 (Cont.).
LINDO OUTPUT

18	0.685	0.000	0.000	0.000	0.000

ROW	ERANDY	FRANDY	GRANDY	HRANDY	IRANDY
1	0.033	0.000	0.000	0.071	0.069
2	1.000	1.000	0.000	1.000	0.069
3	-0.967	0.000	0.000	-0.929	0.000
4	-0.729	0.000	0.000	0.000	0.000
5	-0.967	0.000	0.000	0.000	0.000
6	-0.967	-1.000	0.000	-0.929	0.000
7	1.000	1.000	1.000	1.000	1.000
8	0.000	0.000	0.000	0.000	0.000
9	0.967	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.729	0.000	0.000	0.000	0.000
13	0.967	1.000	0.000	0.929	0.000
14	-1.000	-1.000	0.000	-1.000	-1.000
15	0.967	0.000	0.000	0.929	0.000
16	0.000	0.000	0.000	0.000	0.931
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	JRANDY	KRANDY	SLK 2	SLK 3	SLK 4
1	0.000	0.031	0.000	0.000	0.327
2	0.000	1.000	1.000	0.000	0.000
3	0.000	-0.969	0.000	1.000	1.327
4	0.000	-0.731	0.000	0.000	1.000
5	0.000	-0.969	0.000	0.000	1.327
6	0.000	-0.969	0.000	1.000	1.327
7	0.000	1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.000	0.969	0.000	0.000	-1.327
10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	0.000	0.969	0.000	-1.000	-1.327
14	0.000	-1.000	0.000	0.000	0.000
15	0.000	0.969	0.000	0.000	-1.327
16	0.000	0.000	0.000	0.000	0.000
17	1.000	0.000	0.000	0.000	0.000
18	0.000	0.731	0.000	0.000	0.000

ROW	SLK 5	SLK 6	SLK 7	SLK 8	SLK 9
1	0.000	0.000	0.000	1.000	1.020
2	0.000	0.000	1.000	0.000	0.000
3	1.000	0.000	0.000	1.000	1.020
4	0.000	0.000	0.000	0.000	0.000
5	1.000	0.000	0.000	0.000	1.020
6	1.000	1.000	0.000	1.000	1.020
7	0.000	0.000	1.000	0.000	0.000
8	0.000	0.000	0.000	-1.000	0.000
9	0.000	0.000	0.000	0.000	-1.020

FIGURE 7 (Cont.).
LINDO OUTPUT

10	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	-1.000	0.000	0.000	-1.000	-1.020
14	0.000	0.000	-1.000	0.000	0.000
15	-1.000	0.000	0.000	0.000	-1.020
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	SLK 10	SLK 11	SLK 12	SLK 13	SLK 14
1	1.028	1.036	1.028	1.053	1.031
2	0.000	0.000	0.000	0.000	1.031
3	1.028	1.036	1.028	0.000	0.000
4	0.775	0.781	0.775	0.000	0.000
5	1.028	1.036	1.028	0.000	0.000
6	1.028	1.036	1.028	1.053	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	-1.028	-1.036	-1.028	0.000	0.000
10	-0.775	0.000	0.000	0.000	0.000
11	0.000	-0.781	0.000	0.000	0.000
12	0.000	0.000	-0.775	0.000	0.000
13	-1.028	-1.036	-1.028	-1.053	0.000
14	0.000	0.000	0.000	0.000	-1.031
15	-1.028	-1.036	-1.028	0.000	0.000
16	0.000	0.000	0.000	0.000	0.000
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000

ROW	SLK 15	SLK 16	SLK 17	SLK 18	
1	1.020	0.990	1.020	1.020	-7429.668
2	0.000	0.990	1.020	0.000	97.357
3	1.020	0.000	0.000	1.020	9.814
4	0.000	0.000	0.000	0.769	292.937
5	0.000	0.000	0.000	1.020	1338.793
6	1.020	0.000	0.000	1.020	1112.971
7	0.000	0.000	1.020	0.000	623.673
8	0.000	0.000	0.000	0.000	320.000
9	0.000	0.000	0.000	-1.020	101.207
10	0.000	0.000	0.000	0.000	124.031
11	0.000	0.000	0.000	0.000	125.000
12	0.000	0.000	0.000	0.000	744.186
13	-1.020	0.000	0.000	-1.020	327.028
14	0.000	0.000	-1.020	0.000	1025.811
15	-1.020	0.000	0.000	-1.020	1110.186
16	0.000	-0.990	0.000	0.000	316.832
17	0.000	0.000	-1.020	0.000	816.326
18	0.000	0.000	0.000	-0.769	153.846

APPENDIX A.

SOFTWARE SUMMARY

Data preparation for this project was completed with the use of three pascal routines. These routines started with three input matrices: the Skills Matrix, the Functions Matrix, and the Tasks Matrix. The programs convert these matrices into the Linear programming model used as input into the LINDO software package on the Portland State University computer system. These routines, LPM, LPM2, and FILEPREP, are described below. In addition, a sample input file has been included for reference.

PROGRAM LPM:

This program formulates the efficiency matrix from the skills, function, and tasks matrices. Input is from file INPUT.DAT. See sample file for format. The output is in INPUT.LPM, which is used as input to LPM2.COM. The output contains the number of engineers and tasks, the engineer names, the task names, efficiency matrix, engineer time matrix, and the task time matrix.

PROGRAM LPM2:

This program inputs the number of engineers and tasks, the engineer names, the task names, efficiency matrix, engineer time matrix, and the task time matrix. Input is taken from file INPUT.LPM.

The C matrix is calculated and then the LINDO form of the Linear Programming Model is outputed to file OUTPUT.LPM.

The output is then further formatted by the program FILEPREP.COM.

PROGRAM FILEPREP:

This program reformats the "OUTPUT.LPM" file to ensure that it has 80 bytes in each line. This is a requirement for input using the TAKE command in LINDO. The output is written to READY.LPM. This is the file transferred to the Portland State University computer system.

```
PROGRAM LPM(INFILE,OUTFILE);
{  This program formulates the efficiency matrix from the }
{  skills, function, and tasks matrices.  KS 6/5/88       }
{  Input:   INPUT.DAT     (see sample file for format)    }
{  Output:  INPUT.LPM     (used as input to LPM2.COM )    }
{           The output contains the number of engineers   }
{           and tasks, the engineer names, the task names,}
{           efficiency matrix, engineer time matrix, and  }
{           the task time matrix.                         }


  CONST

    MAXEL = 20;

  TYPE
    NAME     = STRING[15];
    MNAME    = ARRAY [1..MAXEL] OF NAME;
    MATVAR1  = ARRAY [1..MAXEL] OF REAL;
    MATVAR2  = ARRAY [1..MAXEL,1..MAXEL] OF REAL;



  VAR
    INFILE,OUTFILE                  : TEXT;
    SKILLNM,ENGNM,FUNCTNM,TASKNM : MNAME;
    SKILL,FUNCT,TASK,RESULT       : MATVAR2;
    ETIME,TTIME                     : MATVAR1;
    NSKILL,NFUNCT,NTASK,NENG,I,J : INTEGER;



  PROCEDURE GET_NAME(N       : INTEGER;
                VAR  MATNM   : MNAME );

    VAR
      I         : INTEGER;


    BEGIN

      FOR I:=1 TO N DO
        READLN (INFILE,MATNM[I]);
    END;


  PROCEDURE GET_MATRIX1 ( N           : INTEGER;
            VAR           MATRIX      : MATVAR1 );

    VAR
      I         : INTEGER;


    BEGIN
      FOR I:=1 TO N DO
          READ (INFILE,MATRIX[I]);
    END;
```

```
PROCEDURE GET_MATRIX2 ( R,C          : INTEGER;
        VAR             MATRIX       : MATVAR2 );
  VAR
    I,J          : INTEGER;

  BEGIN
    FOR I:=1 TO R DO
      FOR J:=1 TO C DO
        READ (INFILE,MATRIX[I,J]);
  END;



PROCEDURE GET_DATA;

  BEGIN

    READLN (INFILE,NSKILL,NENG,NFUNCT,NTASK);
    GET_NAME (NENG,ENGNM);
    GET_NAME (NTASK,TASKNM);
    GET_MATRIX2 (NENG,NSKILL,SKILL);
    GET_MATRIX2 (NSKILL,NFUNCT,FUNCT);
    GET_MATRIX2 (NFUNCT,NTASK,TASK);
    GET_MATRIX1 (NENG,ETIME);
    GET_MATRIX1 (NTASK,TTIME);

  END;

PROCEDURE MATRIX_MULT (X,Y                    : MATVAR2;
                       XROW,XCOL,YCOL         : INTEGER;
                  VAR  R                      : MATVAR2);

  VAR
    I,J,K          : INTEGER;

  BEGIN

    FOR I:=1 TO XROW DO
      FOR J:=1 TO YCOL DO
        BEGIN
          R[I,J] := 0;
          FOR K := 1 TO XCOL DO
            R[I,J] := R[I,J]+X[I,K]*Y[K,J];
        END;
  END;


PROCEDURE MATRIX_NORM (X                      : MATVAR2;
                       XROW,XCOL              : INTEGER;
                  VAR  R                      : MATVAR2);

  VAR
    I,J,K          : INTEGER;
    SUM            : REAL;

  BEGIN

    SUM:=0;
```

```
    FOR I:=1 TO XROW DO
      FOR J:=1 TO XCOL DO
        SUM:=SUM+X[I,J];
    FOR I:=1 TO XROW DO
      FOR J:=1 TO XCOL DO
        R[I,J]:=X[I,J]/SUM;
  END;


PROCEDURE  DUMP_MATRIX ( X                    : MATVAR2;
                         R,C                  : INTEGER);
   VAR
    I,J         : INTEGER;

  BEGIN
    WRITELN (OUTFILE,NENG,' ',NTASK);

    FOR I:=1 TO R DO
     WRITELN (OUTFILE,ENGNM[I]);
    FOR I:=1 TO C DO
     WRITELN (OUTFILE,TASKNM[I]);

    FOR I:=1 TO R DO
    BEGIN
      WRITE (OUTFILE,X[I,1]*100.0:5:2);
      FOR J:=2 TO C DO
        WRITE (OUTFILE,' ',X[I,J]*100.0:5:2);
      WRITELN (OUTFILE);
    END;
    FOR I:=1 TO R DO
      WRITELN (OUTFILE,ETIME[I]);
    FOR I:=1 TO C DO
      WRITELN (OUTFILE,TTIME[I]);


  END;


BEGIN

  ASSIGN (INFILE,'INPUT.DAT');
  ASSIGN (OUTFILE,'INPUT.LPM');
  RESET (INFILE);
  REWRITE (OUTFILE);


  GET_DATA;
  MATRIX_NORM (SKILL,NENG,NSKILL,SKILL);
  MATRIX_NORM (FUNCT,NSKILL,NFUNCT,FUNCT);
  MATRIX_NORM (TASK,NFUNCT,NTASK,TASK);

  MATRIX_MULT (SKILL,FUNCT,NENG,NSKILL,NFUNCT,RESULT);
  MATRIX_NORM (RESULT,NENG,NFUNCT,RESULT);
  MATRIX_MULT (RESULT,TASK,NENG,NFUNCT,NTASK,RESULT);
  MATRIX_NORM (RESULT,NENG,NTASK,RESULT);
  DUMP_MATRIX (RESULT,NENG,NTASK);
  CLOSE (INFILE);
  CLOSE (OUTFILE);
```

```
END.
```

```
PROGRAM LPM2(INFILE,LPM);

{       This program inputs the number of engineers and
        the number of tasks.  Then inputs the names of
        the engineers and the tasks.  Following this, the
        Efficiency matrix is inputed along with the
        engineer time and task time matrices.

        The C matrix is calculated and then the LINDO form
        of the Linear Programming Model is outputed.

        The output is then further formatted by the
        program FILEPREP.COM.


        Input:    INPUT.LPM
        Output:   OUTPUT.LPM   (used as input to FILEPREP.COM)
}



  VAR
    INFILE,LPM,CMATRIX              : TEXT;
    NENG,NTASK                      : INTEGER;


PROCEDURE GO;

  TYPE
    NAME     = STRING[15];
    ENAME    = ARRAY [1..20] OF NAME;
    TNAME    = ARRAY [1..20] OF NAME;
    TEMATVAR = ARRAY [1..20] OF REAL;
    TTMATVAR = ARRAY [1..20] OF REAL;
    MATVAR2 = ARRAY [1..20,1..20] OF REAL;

  VAR
    ENGNM                           : ENAME;
    TASKNM                          : TNAME;
    TASK,RESULT                     : MATVAR2;
    ETIME                           : TEMATVAR;
    TTIME                           : TTMATVAR;
    I,J                             : INTEGER;
    SUM                             : REAL;



  BEGIN
    BEGIN

      FOR I:=1 TO NENG DO
       BEGIN
```

```
      READLN (INFILE,ENGNM[I]);
    END;

END;
BEGIN

  FOR I:=1 TO NTASK DO
   BEGIN
    READLN (INFILE,TASKNM[I]);
   END;

END;

BEGIN
  FOR I:=1 TO NENG DO
    FOR J:=1 TO NTASK DO
     BEGIN
      READ (INFILE,TASK[I,J]);
     END;
END;

BEGIN
  FOR I:=1 TO NENG DO
    BEGIN
      READ (INFILE,ETIME[I]);
    END;
END;

BEGIN
  FOR I:=1 TO NTASK DO
    BEGIN
      READ (INFILE,TTIME[I]);
    END;
END;

BEGIN

  FOR I:=1 TO NTASK DO
   BEGIN
    SUM := 0;
    FOR J:=1 TO NENG DO
       SUM:=SUM+TASK[J,I];
    FOR J:=1 TO NENG DO
      RESULT[J,I]:=TASK[J,I]/(SUM/NENG);
   END;
END;



WRITE (LPM,'MIN ',TASKNM[1],ENGNM[1]);
FOR I:=1 TO NENG DO
  BEGIN
   FOR J:=1 TO NTASK DO
    IF NOT((I=1) AND (J=1)) THEN
     BEGIN
      IF  ((((I-1)*NENG)+J) MOD 4)=0 THEN WRITELN (LPM);
      WRITE (LPM,'+',TASKNM[J],ENGNM[I]);
     END;
  END;
  WRITELN (LPM);
```

```
    WRITELN (LPM,'ST');

  FOR I:=1 TO NENG DO
    BEGIN
     WRITE (LPM,TASKNM[1],ENGNM[I]);
     FOR J:=2 TO NTASK DO
       BEGIN
        IF (J MOD 4)=0 THEN WRITELN (LPM);
        WRITE (LPM,'+',TASKNM[J],ENGNM[I]);
       END;
     WRITELN (LPM,' < ',ETIME[I]:8:2);
    END;

  FOR J:=1 TO NTASK DO
    BEGIN
     WRITE (LPM,RESULT[1,J]:5:2,TASKNM[J],ENGNM[1]);
     FOR I:=2 TO NENG DO
       BEGIN
        IF (I MOD 4)=0 THEN WRITELN (LPM);
        WRITE (LPM,'+',RESULT[I,J]:5:2,TASKNM[J],ENGNM[I]);
       END;
     WRITELN (LPM,' > ',TTIME[J]:8:2);
    END;
    WRITELN (LPM,'END');
    WRITELN (LPM,'LEAVE');

  END;



BEGIN

   ASSIGN (INFILE,'INPUT.LPM');
   ASSIGN (LPM,'OUTPUT.LPM');
   RESET (INFILE);
   REWRITE (LPM);
   READLN (INFILE,NENG,NTASK);
   GO;
   CLOSE (INFILE);
   CLOSE (LPM);

END.
```

```
PROGRAM FILEPREP (INFILE,OUTFILE);

      This program reformats the "OUTPUT.LPM" file to
      ensure that it has 80 bytes in each line.  This
      is a requirement for input using the TAKE command
      in LINDO. The output is in READY.LPM.  This is
      the file transferred to the Portland State
      University computer system.
}


  VAR
   CH          : CHAR;
   INFILE      : TEXT;
   OUTFILE     : TEXT;
   COUNT       : INTEGER;


  BEGIN
    ASSIGN (INFILE,'OUTPUT.LPM');
    ASSIGN (OUTFILE,'READY.LPM');
    RESET (INFILE);
    REWRITE (OUTFILE);
    WHILE NOT EOF(INFILE) DO
      BEGIN
        COUNT:=0;
        WHILE NOT EOLN(INFILE) DO
          BEGIN
            READ(INFILE,CH);
            WRITE(OUTFILE,CH);
            COUNT:=COUNT+1;
          END;
        WHILE COUNT<80 DO
          BEGIN
            WRITE(OUTFILE,' ');
            COUNT:=COUNT+1;
          END;
        READLN(INFILE);
        WRITELN(OUTFILE);
      END;
      CLOSE (INFILE);
      CLOSE (OUTFILE);
   END.
```

SAMPLE INPUT DATA FILE

NOTE: Blank lines and comments have been added for purposes of this example only.

```
9 6 7 11                    (s SKILLS, e ENGINEERS, f FUNCTIONS, t TASKS)

SAM                         (ENGINEER'S NAMES)
STEVE
KEN
JOHN
PAUL
RANDY

A                           (TASK IDENTIFICATIONS)
B
C
D
E
F
G
H
I
J
K

2.30  1.15  1.72  2.30  1.72  1.44  0.57  2.30  2.01    (Skills Matrix)
2.01  1.15  2.30  1.72  1.72  2.59  0.57  0.86  2.59
2.59  2.59  2.30  2.30  2.30  1.72  2.30  2.59  2.59
1.72  2.01  1.44  2.59  2.59  1.72  2.01  0.57  1.44
1.44  2.01  2.59  2.30  2.01  1.44  1.72  0.57  1.44
2.30  2.30  2.01  2.01  1.72  0.57  2.30  2.30  0.57

2.11  1.85  1.85  0.53  0.79  0.79  2.37              (Functions Matrix)
1.85  1.58  1.32  0.53  0.79  1.06  2.11
1.58  1.85  2.11  1.32  1.32  0.79  2.37
0.53  1.32  1.85  2.37  2.11  1.85  0.53
0.53  1.85  1.85  2.37  1.85  1.85  0.53
2.11  1.58  1.85  2.37  2.37  2.37  1.32
2.11  1.85  1.06  0.53  0.53  1.58  1.85
1.32  1.32  1.85  2.37  2.37  2.37  1.32
2.37  1.85  1.06  1.06  1.06  1.58  2.37

8.11  4.50  2.70  0.90  0.90  0.00  0.00  0.00  0.00  1.80  1.80  (Task Matrix)
0.90  4.50  6.31  7.21  4.50  1.80  1.80  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.90  4.50  1.80  6.31  1.80  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  5.41  0.90  6.31  1.80  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.90  6.31  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.90  0.00  5.41
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  7.21  1.80

1440                        (TIME AVAILABLE FOR EACH ENGINEER)
1440
1440
1440
1440
1440
```

SAMPLE INPUT DATA FILE (Cont.).

```
320                    (EFFECTIVE ENGINEERING TIME REQUIRED FOR TASKS)
480
160
160
960
320
1600
2400
320
800
200
```