US Army Corps
of Engineers
Construction Engineering
Research Laboratory

USACERL Technical Report P-92/14
February 1992
VIC-Based Engineer FAM (AMIP)

Representing the Combat Engineer Function in Land Combat Models: Lessons Learned

by
Carol Subick

The Engineer Model Improvement Program (EMIP) was established in 1988 as a comprehensive effort to ensure that engineers are properly represented in the Army's land combat models. The focus of EMIP was to enhance the engineer representation in the Vector-In-Commander (VIC) battle simulation. During the past 2 years, a new engineer module was designed and developed to replace VIC's original representation of the combat engineer functional area.

This report documents the lessons learned about modeling the combat engineer functional area, combining the important elements of the VIC project with U.S. Army Construction Engineering Research Laboratories experience from several earlier engineer enhancement efforts. The lessons presented here concern land combat models in general as well as combat engineer modeling.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products. The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED

DO NOT RETURN IT TO THE ORIGINATOR

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE February 1992	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE

Representing the Combat Engineer Function in Land Combat Models:
Lessons Learned

5. FUNDING NUMBERS

PR AT41
TA SE
WU Y21

6. AUTHOR(S)

Carol A. Subick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Construction Engineering Research Laboratory (USACERL)
P. O. Box 9005
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER

TR P-92/14

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

US Army Engineer School
ATTN: ATSE-CDC-M
Fort Leonard Wood, MO 65473

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161

12a. DI BUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Engineer Model Improvement Program (EMIP) was established in 1988 as a comprehensive effort to ensure that engineers are properly represented in the Army's land combat models. The focus of EMIP was to enhance the engineer representation in the Vector-In-Commander (VIC) battle simulation. During the past 2 years, a new engineer module was designed and developed to replace VIC's original representation of the combat engineer functional area.

This report documents the lessons learned about modeling the combat engineer functional area, combining the important elements of the VIC project with U.S. Army Construction Engineering Research Laboratories experience from several earlier engineer enhancement efforts. The lessons presented here concern land combat models in general as well as combat engineer modeling.

14. SUBJECT TERMS

Engineer Model Improvement Program
Vector-In-Commander battle simulation model

15. NUMBER OF PAGES

120

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

FOREWORD

This work was performed for the Office of the Chief of Engineers (OCE) under Project 4A162734AT41, "Military Facilities Engineering Technology"; Task, SE; Work Unit Y21, "VIC-Based Engineer FAM (AMIP)." The Technical Monitor was MAJ Dave Davis, ATSE-CDC-M, Army Engineer School, Fort Leonard Wood, MO.

The research was conducted by the Facility Systems Division (FS), U.S. Army Construction Engineering Research Laboratories (USACERL). Dr. Michael J. O'Connor is Chief of USACERL-FS. The USACERL technical editor was Gloria J. Wienke, Information Management Office.

COL Daniel Waldo, Jr., is Commander and Director of USACERL, and Dr. L.R. Shaffer is Technical Director.

CONTENTS

CONTENTS (Cont'd)

REPRESENTING THE COMBAT ENGINEER FUNCTION IN LAND COMBAT MODELS: LESSONS LEARNED

1 INTRODUCTION

Background

During the past 30 years, the Army has placed increasing reliance on computer combat simulations to answer difficult questions regarding force structure, doctrine, and material acquisition. Much of the research and development effort associated with land combat models has been devoted to representing how units fight. But the Army's success on the modern battlefield does not depend solely on the lethality of weapons; it depends on effective maneuver, correct use of the terrain, protection of vital assets, and proper coordination with combat support and combat service support functions. Building an accurate combat model requires a realistic portrayal of such important factors. Since combat engineers greatly influence all these factors, a combat model's representation of the engineer functional area is crucial to the validity of the decisions to be drawn from the model.

Most of the land combat models developed during the past 30 years have included, at best, only a minimal representation of the combat engineer function. During the 1960's and 1970's, this may have been because models were developed for specific studies and may have actually required only minimal engineer representations. When the Army Model Improvement Program (AMIP) was instituted in 1980 to control model proliferation, however, the neglect of the engineer functional area began to have serious consequences. The program limited the number of models to be developed and placed on each surviving model the burden of providing a realistic battlefield environment in which to study a broad range of questions. The standing models accredited for use in major studies were unable to capture the combat engineer contribution to the combined arms force.

During the 1980's the engineer community attempted to rectify the situation by enhancing the combat engineer representations of the AMIP models already in use. The U.S. Army Engineer School (USAES), the modeling proponent for engineers, established the Engineer Modeling Program to address the three models in the AMIP hierarchy: Combined Arms and Support Task Force Evaluation Model (CASTFOREM), Corps/Division Evaluation Model (CORDIVEM), and Force Evaluation Model (FORCEM). The U.S. Army Construction Engineering Research Laboratories (USACERL) was tasked to develop new combat engineer representations for CORDIVEM and FORCEM.

The focus of the USACERL efforts on both CORDIVEM and FORCEM was each model's representation of combat engineers themselves and of the processes they use to perform their function on the battlefield. The size and complexity of CORDIVEM and FORCEM, combined with the fact that both models were already close to being fully operational, precluded any attempt to change the way either model portrayed the effects of the engineer activity on the other elements of the force. This would have been a critical deficiency if both models had continued to fulfill their roles in the AMIP hierarchy. In 1986, CORDIVEM was replaced by a new model, Vector-In-Commander (VIC), just 2 months after the newly developed engineer module was installed in it. FORCEM has not reached its full study potential, and its new engineer module has never been used. When VIC replaced CORDIVEM, the engineer community was essentially back at the starting position with another Corps-level model with an inadequate representation of the combat engineer function.

Part of the problem was that no general representation of the combat engineer function had been designed; each model enhancement program had been peculiar to the given model. Consequently, USACERL began working with USAES in 1986 to identify the minimal requirements using an in-house model called the Force Structure Trade-off Analysis Model (FSTAM). FSTAM was to be a testbed, a model with which engineer analysts at USAES could experiment to determine which engineer elements were essential, which were not, which were missing, and which were properly or improperly played. In turn, model designers at USACERL would use the feedback to gradually refine FSTAM into an engineer functional area model (EFAM) and ultimately produce a standard design for an engineer representation at the Corps level.

In its role as a testbed, FSTAM had to satisfy a number of operational requirements. Rapid turnover of a limited staff at the Army Engineer School and the lack of sophisticated computer equipment and the personnel to operate it led to an early design decision to implement the model in a microcomputer environment. Additionally, ease of use became a prime objective. To that end, FSTAM was designed to display an animated graphical representation of the battlefield, to have companion menu-driven processors for scenario development and management, and to offer a mouse-driven graphical approach to digitizing the terrain data base. The work on these elements outside the engineer representation exposed researchers at USACERL to the larger framework of the modeling environment, including the management of scenarios and the actual use of a model for analysis. This exposure added a new perspective, forcing the researchers to see the importance of a natural, intuitive design for a model.

Plans for using FSTAM were overtaken by events, primarily the move of the Army Engineer School from Fort Belvoir, VA to Fort Leonard Wood, MO and the development of the Engineer Model Improvement Program (EMIP). This program focused attention on enhancing VIC and making it the engineer functional area model. In support of that effort, USACERL began to develop a new engineer module for VIC in 1988 and completed the task in 1991.

Though USACERL has documented each model's improved engineer representation, personnel changes and the press of new projects have prevented documenting the more general lessons learned from each model enhancement. This report identifies the common elements in all the models to date and establishes a base case of model requirements upon which to build as experience grows.

Objective

The objective of this report is to document the lessons learned about modeling the combat engineer functional area, combining the important elements of the VIC project with USACERL's experience from several earlier enhancement efforts.

Approach and Scope

USACERL's experience with representing the engineer functional area in land combat models spans 10 years and four major model developments. This report addresses the following three areas arising from this experience that are of interest to the combat modeling community:

1. General observations about combat modeling and the problems associated with the development of standing models,

2. Elements of the combat engineer functional area that are fundamental to representing the function in a battle simulation, and

3. The essential elements of a combat engineer representation and a proposal for an engineer module design structure.

This report assumes that the reader is familiar with the design and use of land combat models for analysis.

2 SIMULATION MODELING AND THE STUDY OF LAND COMBAT

Definition of Terms

The term "model" is generally defined to be any representation of an entity in a form other than that of the entity itself. The entity may be an object, a system, or even a concept. So the word model covers a very large universe of applications.

Such an all-inclusive definition for the term model leads to some confusion about the more specific term "simulation." A. Alan Pritsker, in a review of books and journal articles on the subject of simulation, found over twenty different views of the meaning of the word.[1] Some analysts identify the model with the simulation; some say simulation is the process of using the model; some say it is the process of designing and using the model; and some say simulation is simply an analytic technique. In addition, some analysts debate whether computers play a role or whether the processes being simulated must involve some element of randomness or must be modeled numerically.

As a starting point for this report, the author adopted the definition of simulation offered by Pritsker.

> If a system can be characterized by a set of variables, with each combination of variable values representing a unique state or condition of the system, then manipulation of the variable values simulates movement of the system from state to state. This is precisely what simulation is: The representation of the dynamic behavior of the system by moving it from state to state in accordance with well-defined operating rules.[2]

This definition captures the key concept that a simulation is a representation of how a system changes over time. But even this definition illustrates some of the confusion involved in drawing precise boundaries around the word so one may properly decide what is and is not included. Clearly, a simulation is a specific type of model. But the set of variables that characterize the static state of the system is also a model of the system, and the definition is not clear about whether that static model is part of the simulation. In addition, computer simulations represent the system and its dynamic behavior in code and data but only move the system from state to state when actually run. So it is easy to see why so many conceptual views of the word simulation exist. This report will make no attempt to be precise in definitions and use; the words "simulation" and "model" will be used interchangeably and the term "simulation model" will be used to refer to the underlying structure of the objects, relationships, and processes that represent the tangible elements of the system.

Pritsker's definition is appropriate for the more specific case of discrete event simulations, which are the standard representational paradigm for land combat. In a discrete event simulation, changes in the state of the system take place instantaneously at specified moments in time as determined by the rules of interaction between system components.

Designing a discrete event simulation of a complex system requires so much simplification and abstraction that an immediate concern must be the "goodness of fit" between the model and the system it represents. Measures of the accuracy of a simulation and its results are easy to formulate when the represented system can be manipulated for comparison. Both the model builder, who focuses on the internal structures, and the model user, who focuses on the input/output, can experiment with the

[1] A.A.B. Pritsker, "Compilation of Definitions of Simulation," *Simulation*, August 1979, pp 61-63.

[2] A.A.B. Pritsker, and C.D. Pegden, *Introduction to Simulation and Slam* (Halsted Press, New York, 1979), p 6.

represented system to measure the accuracy of model functions and results. Land combat is not such a manipulable system, however, and measures of the accuracy of combat simulations do not exist.

The term "realistic representation" is often used in describing what modelers hope to achieve with combat simulations, but the term is imprecise. Combat model builders use the term when they are confident that the internal structures of their model properly take into account all of the significant objects and interactions on the battlefield. Combat model users apply the term to a model when military experts agree that the model's outputs are consistently close to what they would expect to happen on a real battlefield. For a system as complex as land combat, the most that can be hoped for as a measure of goodness of fit is a level of confidence that the model will not be misleading. In this regard, the term "valid" is used to describe a model in which there is a high level of confidence that the insights gained from it are correct.

The Structure of a Simulation Model

A simulation model consists of six components, though the actual structure of the implementation may not clearly identify some of them. They are:

1. The representation of the relevant objects of the modeled system, including their attributes and their relationships to one another,

2. The representation of each event that changes the state of the modeled system, i.e., procedures for determining what happens to the objects, their attributes, and their relationships when a certain event occurs,

3. System data—data that determines the initial state of those attributes of system objects that are natural to the objects and do not tend to change from one run of the simulation to another,

4. Scenario data—data that determines the initial state of those attributes of system objects that are peculiar to a specific run of the simulation,

5. A decision mechanism to determine what happens next; i.e., what new events will occur because of a change in the system, and

6. A simulation controller to handle the initialization of the state of the system and to manage the flow of action between event occurrences and decision processes.

In simulating a chess game, for example, the system objects are the playing pieces, the board, and the opposing players; attributes of a playing piece might include its type and its location on the board; and relationships might link each playing piece to its player/owner and to the rules governing its legal moves. System data sets the attributes of the playing pieces at the start of the game, while the scenario data might state which player has the first move and perhaps what the move is. One event would be the movement of a piece from one position to another; its processes would include relocating the moved piece and perhaps removing a captured piece. After such an event, the opposing player's decision mechanism would be consulted to determine and schedule the next move. As computer chess games are usually set up, the decision mechanism is split between a human player entering moves interactively and an automated player whose competence can be adjusted externally. The simulation manager keeps track of the positions of all of the pieces on the board, controls the progress of the game from one move to another, and perhaps determines when checkmate has been achieved to end the game.

In a Simscript II.5 computer simulation, the system objects (entities), their attributes, and their set relationships are specified in the preamble. The events are identified in the preamble and given procedurally in the code. System data and scenario data are input data, sometimes entered interactively and sometimes by file processing. Generally, no distinction is made between the two types of data. The simulation controller is in the code, facilitated by the fact that Simscript II.5 is a simulation programming language with a built-in timing mechanism and event processor. The decision mechanism can be implemented in many ways, even within a single model. Indeed, each decision may be the result of a unique process.

Though this model structure identifies six components, a computer simulation is fundamentally data and code. The system and scenario data are obviously data, while the representations of objects and events and the simulation manager are naturally expressed in code. The decision mechanism, however, does not fit neatly into either category. Of the six simulation components, the decision mechanism is the one that designers and programmers have the most difficulty implementing. Ironically, the validity of the simulation's dynamics is totally dependent on how well the decision mechanism works.

Programming languages have always handled data manipulation and mathematical calculation with more facility than they have handled concepts and logical reasoning. The structured approach associated with these languages encourages model designers to think in terms of sequential steps. In this environment, the "what happens next?" question of the decision mechanism is usually easier to handle as a numeric calculation and comparison than as a conceptual analysis of the situation. The typical older simulation relies on hard-coded procedures for the bulk of its low-level decisions and allows only an occasional input from an interactive human or an automated system of decision tables to guide the high-level decisionmaking. The result is that the decision mechanism is too rigid to adequately reflect the dynamics of the system; decisions that should be under the control of the user are out of reach unless the code is changed.

With new tools, especially those coming from the fields of object-oriented analysis/design/programming and artificial intelligence, techniques for handling these difficulties are evolving. These techniques offer new approaches to analyzing the system to be simulated and improved methods for implementing the simulation as code and data. In particular, the object-oriented approach of focusing on objects and their behaviors rather than on system processes provides a powerful strategy for analyzing the dynamics of the real system and for overcoming some of the problems associated with the decision mechanism component of the simulation.

This strategy starts with the idea that the objects being simulated are subject to two types of behaviors: **natural behaviors** resulting from their real-world physical characteristics and **controlled behaviors** resulting from decisions determined by a controlling authority. Controlled behaviors cannot override natural behaviors but are likely to be subject to the restrictions of natural behaviors. For example, a car consumes measured quantities of fuel as it moves and limits its speed to a predetermined range as natural behaviors; but where the car goes and how fast it travels are controlled behaviors. Also, the ability of the car to actually go somewhere is subject to its natural behaviors to require fuel to get there and to limit its speed to a specified maximum.

Important connections exist between natural behaviors, system data, and the simulation controller and between controlled behaviors, scenario data, and the decision mechanism. Essentially, the system data about objects determines parameters for their natural behaviors, and the simulation controller uses natural behaviors as part of its management of the flow of action. On the other hand, scenario data about objects determines parameters for their controlled behaviors, and the decision mechanism uses controlled behaviors as part of its processing scheme.

By thinking in this way about the system to be simulated, easier distinctions are made between what processes/behaviors should be coded as a part of the simulation controller and what processes/behaviors should be left to the decision mechanism. Natural behaviors can be represented entirely in code, while controlled behaviors should be linked to input facts and an automated reasoning system.

Lesson Learned: Maximum flexibility and robustness in model design can be achieved when every behavior that can be controlled in the real world system is mapped to a behavior that can be controlled by the model user in the simulation.

Land Combat Models

The process of designing a simulation begins with (1) identifying the significant objects and the properties of those objects that define the state of the system being modeled, (2) understanding the interactions of those objects that cause the state of the system to change, and (3) expressing those interactions in well-defined operating rules compatible with the simulation's implementation.

The system at hand is combat, and combat is not well understood. Indeed, one might question whether one can understand it at all or whether the formulation of combat's "well-defined operating rules" is possible. In the best of circumstances, the models of combat to date require a "leap of faith" to trust in their overall results, and they are not accurately predictive in any of their individual details. Even so, such models have provided valuable insights to military analysts and have proved their worth despite their imperfections. So the intent here is not to question the merit of using simulations for combat analysis nor to suggest that current combat simulations are of little merit because they are not accurately predictive, but to suggest that the analytic quality of combat simulations can be improved by changing the way they are developed.

A basic premise of this report is that the quality of a combat simulation and any insights that may be drawn from it directly depend on how well the designer performs the three steps listed above. Most of the problems stem from identifying the wrong objects or attributes as significant and representing object interactions incorrectly. A usual approach to modeling land combat has been to represent what happens on the battlefield in a superficial way and to rely on including a large number of details to establish claims of a realistic representation. Only rarely do combat models capture the concepts behind the essential cause and effect relationships that lead to the simulated action.

The use of computers, with their capacity for handling immense amounts of data and complex calculations, has only encouraged the tendency to add more detail to simulations. Advances in hardware and software technology have led to the growth of such large, complex combat simulations that their validity cannot be established even at the grossest levels of confidence. Recently, much of the concern about the validity of these models has addressed the computer-related problems, including the instability of floating-point arithmetic and the problems inherent in the sequential representation of a world of parallel events. Simulation designers must be careful to construct models whose results are as independent as possible of the characteristics of a machine's arithmetic operations and the order of processing simultaneous events. But USACERL's experience indicates that the model design process itself, independent of computer implementation, is the area where a combat model's validity is most at risk.

The ability to build a valid model, or even more so to prove the validity of an existing model, shrinks as the complexity of the system increases. Each expansion of a model to include new objects and new interactions strikes at one's confidence that the model continues to provide a satisfactory representation of the real-world system. The best approach to using simulation for analysis is to design a unique, well-focused simulation for each study question, limiting the number and types of objects represented to the smallest possible domain for the question at hand. Combat simulations should fall under this rule of thumb.

Unfortunately, the time and expense required to develop a specific model for each military study question is prohibitive. The Army relies on the use of a few standing models of combat to address a variety of questions rather than develop a new model for each study. Even removing the time and money constraints would not eliminate the need for such multipurpose standing models. The strong interrelationships that exist between the different battlefield components usually make it impossible to limit a model's domain to a small number of factors. Models specifically designed for different studies ultimately duplicate the same basic structures and representations.

In addition, many study questions involve trade-off analysis, force restructuring, or doctrinal development. These types of questions actually require the features of a full-scale combat model—a model that includes representations of most, if not all, of the functional elements used in the full depth of the battle area and that provides an environment for studying a wide range of alternatives. During the early 1990's, the most prevalent study questions will focus on reorganizing and downsizing of the Army and shifting the strategy from a forward deployed defense along a linear front to a projection of land combat power from the continental United States to a nonlinear battlefield.

To accommodate a wide range of study questions, the Army's standing combat models are designed to provide a realistic battlefield environment in which to test a variety of alternative actions. That requirement automatically forces standing models to a level of complexity at which a single person is unable to retain a complete mental picture of all of the model's functions and relationships. Problems abound when this level of complexity is reached. A group of experts in military modeling addressed this issue in a committee report for the Military Operations Research Society (MORS):

> ... We caution that attempts to design all-purpose models to serve many problems and decisions have rarely succeeded. For one thing, the scope of the undertaking has been consistently underestimated. Since an all-purpose model's intended uses are universal, it must cover all aspects of every potential problem that it may be called upon to address. The bigger and more expensive it becomes, the more problems it must treat in order to justify its expense. In the 1960s especially analysts attempted to build many such models by exploiting the capacity of high-speed computers. Instead of trying to determine in advance which were the important variables, their brute-force method treated all variables as important and assumed that the computer would overpower nature. As usual, nature won, overwhelming the computer and the analyst with her complexity. Even models at the micro level suffered from (1) the tedium of entering inputs; (2) the opacity of the processes; (3) the lack of data commensurate with the detail in the model; (4) the large number of parameters that had to be varied, with concomitant difficulties in determining the parameter combinations to which to give the most weight or credence; and (5) difficulties of corroboration, even though realism, physical fidelity, and validity were the motivations underlying them.[3]

The wisdom of this statement and its implications for the model design process should not be underrated. The two most important lessons learned are very general.

> **Lesson Learned: Massive computer power is not the missing element in the quest to build valid combat models. A poorly designed simulation will not give better results because of improved hardware. (This lesson is an application of two familiar computer maxims: "Garbage in, garbage out," and "Make it work first before you make it work fast.")**

[3] *Military Modeling*, Wayne P. Hughes, Ed. (Military Operations Research Society), 1984, p 22.

Lesson Learned: Limiting the model to include only the important variables is essential to the validity of the model, not just its computer run time. The design must capture the essence of each object and interaction.

A model of an extremely complex system should not be bogged down with inconsequential details or convoluted interactions that allow inconsequential details to have a disproportionate effect. The committee report to MORS lists five problems that may all be direct consequences of the failure to design "lean and mean" models. The effect of the sum of the five is that model users generally do not understand what the model does and are likely to misuse the model or misinterpret its results.

Errors made by including inconsequential details are minor, however, when compared to errors made by omitting essential details.

Lesson Learned: Errors in design associated with including inconsequential details tend to occur at the lower levels of the system representation, while errors in design associated with excluding essential details tend to occur at the higher levels.

Representation of the higher levels of any system tends to be more qualitative than quantitative, so computer modeling techniques enforce the tendencies for high-level errors because of their facility for handling quantitative details over qualitative details. For example, the representation of a combat unit's movement may take into account the effect of elevation, ground cover, weather, visibility, etc., in determining the unit's speed, but may not consider the effect of the commander's original intent for the maneuver. Such a representation ignores what may be the one element that determines how rapidly a unit must move to its new location but performs complicated but imprecise calculations to determine the effect of something like a rise in elevation on unit speed. Model designers use mathematical calculation instead of conceptual reasoning and end up with a distorted representation of "what happens" instead of a more accurate representation that accounts for "why things happen."

The problems with standing models only escalate when you take into account the most likely battlefield environments of the future. Intelligence and electronic warfare, the complexities of command, control, and communications (C^3), long-range strike capabilities, the growing reliance on maneuverability and rapid response, etc., will play increasingly more important roles in determining the outcome of future engagements. These battlefield elements fall more into the area of "why things happen" than into the area of "what happens," and they must be taken into account if models are to have any hope of properly portraying modern land combat.

One might logically conclude that a benefit of using standing models instead of continually designing new study-specific models is that standing models can be refined over time to improve the faithfulness of their representations—both what we see and why we see it—and, therefore, the trustworthiness of their results. A negative aspect of constantly refining standing models is explored in the committee report to MORS:

> Since the same model will often be maintained and operated in different locations by different commands, a system of change orders will be required. Then there are inevitable conflicts between rapid redesign to handle new demands and the need for stability and field personnel to keep model and documentation up-to-date. Computer hardware and technology are advancing so rapidly that hardware and software obsolescence become serious obstacles to model perpetuation.[4]

[4] *Military Modeling.*

The configuration control problems mentioned here have more than a managerial impact. The lack of documentation, the complexity of the model structure, and the seemingly constant need to change the model all combine to produce a situation that is devastating to model integrity. No one person can manage all the changes, and the group of people making changes do so without adequate information about the model's basic structure and the far-reaching consequences of what they are doing to it. Instead of gradually improving the model, the "enhancement" process causes a gradual disintegration.

Army analysts are in a "catch-22" situation with current standing models of combat. For their most prevalent types of study questions, only a full-scale model including representations of all of the major functional areas on the battlefield is suitable. But such models tend to be so large and complex that the very process of developing and using them undermines a willingness to trust the insights gained from them. The way out of this dilemma is to be found at the very beginning of the model design process.

> **Lesson Learned: Since it is almost a given that any standing combat model will be changed many times, the original structure and all changes to it must be such that the change process maintains rather than undermines the soundness of the design. This comes from a carefully crafted original and from a clear understanding of its structure by those who would offer changes to it.**

Success relies on careful design from the outset and careful documentation and configuration control during the model's entire life cycle from original implementation to final study use. The portion of this lesson that is independent of the computer environment is the design of the original structure. This is the starting point and key to the entire process. If it is not done well, all other efforts count for nothing.

The Army Model Improvement Program (AMIP)

In 1980 Army Regulation (AR) 5-11 established the Army Model Improvement Program under the direction of the Army Models Committee to control the development, documentation, and implementation of a hierarchical family of computerized combat models to support Army studies, research, and training. Combat models accredited under AMIP fall into one of three levels of resolution: Combined Arms/Support Task Force, Corps/Division, and Theater. The levels of resolution are tied to length of time simulated, size of the battle area, and the number and size of units involved:

High Resolution: Combined Arms/Support Task Force Level
Length of Time Simulated: 1 to 3 hours
Typical Land Area Involved: 20 km by 20 km
Focus: Operations of item systems
Current AMIP Analytic Model: Combined Arms and Support Task Force Evaluation Model (CASTFOREM)

Mid Resolution: Corps/Division Level
Length of Time Simulated: 3 to 6 days
Typical Land Area Involved: 200 km by 200 km
Focus: Operations of combined arms and support task forces
Current AMIP Analytic Model: Vector-In-Commander (VIC)

Low Resolution: Theater Level
Length of Time Simulated: 3 to 6 months
Typical Land Area Involved: Entire Theater of Operations
Focus: Combined, joint and unilateral theater operations and support of those operations
Current AMIP Analytic Model: Force Evaluation Model (FORCEM)

The models in the AMIP hierarchy are designed to study concepts, tactics, and resource requirements for units at their respective resolution levels. Though each model has its own unique view of the battlefield, the models share a number of common properties. AR 5-11 established Simscript II.5 by CACI as the programming language of choice, and all of the above models are implemented in that language. Additionally, the models are expected to represent every functional area relevant to their scope in a manner consistent with that scope. This ultimately means that all of the models make some attempt to portray most combat, combat support, and combat service support elements, though each model sees the functional areas from a different perspective. In addition, AR 5-11 spells out six general requirements for models at all three levels of resolution:

"a. *Symmetry.* Threat and allied forces must be capable of being treated with the same fidelity accorded U.S. forces recognizing that there are asymmetries in organizations, tactics, and doctrine. Current, projected, and response capabilities will be considered.

b. *Battlefield environment.* The full spectrum of battlefield environmental conditions will be represented. These include terrain, weather, obscurants (such as smoke and dust), contaminants (such as chemical, biological, and radiological), electronic warfare, and nuclear effects.

c. *Human performance factors.* The models will reflect, to the extent feasible, those human factors that affect organization and system performance. These factors include fatigue and level of training.

d. *Transparency.* The models will be clearly documented, provide a clear audit trail from input to output, and assist the user in understanding why outcomes change as inputs are varied.

e. *Flexibility.* The models will apply to various geographic areas and operational scenarios. These will range from a nuclear war or an intense nonnuclear battle in Europe to a contingency operation in another area.

f. *Responsiveness.* To maximize the usefulness of the models and minimize total throughput time, every effort will be made to simplify input data preparation, minimize running time, and facilitate analysis of outputs."

Designing a combat model that fulfills all of these requirements is an extremely difficult task. The modeling community has very little data regarding the effects of factors like weather and soldier fatigue, which vary in significance according to the threat, the geographic location, and the tempo of the battle. Achieving symmetry of representation is difficult even when the threat and likely tactics are known. In the present era, no one can be sure of either who the next threat will be or what tactics will be used against us. After narrowing the list of possible threats and the likely battle doctrines, model designers face wide variances in tactics related solely to the geographic location of the conflict. The number of possible scenarios grows geometrically. In addition to the representational requirements, the model itself must simultaneously satisfy the criteria that it be easy to set up and transparent to use.

On the other hand, having a model fail to meet all of the above requirements raises serious questions about its analytic value, either by limiting the scope of the model's applicability, or by degrading its validity because significant factors have been ignored, or by making the very process of using it too difficult. This is another aspect of the "catch-22" situation. Earlier, the preservation of model validity seemed simultaneously dependent upon and destroyed by incremental expansion and refinement of a model's representations. Here, the logical reaction is to conclude that the model requirements themselves are contradictory, so a model that meets them all will never be built. This is certainly true if the approach to the design is to add more and more detail to the representation of each significant factor. The lesson here expands on the importance of careful design in building the original model structures:

> **Lesson Learned: The only way to build an easy-to-use full-spectrum combat model applicable to a variety of theaters is to use a high degree of abstraction, with the guiding principle of severely limiting detail and basing the flow of action more on logical concepts than on mathematical calculations.**

In the abstraction process, <u>every</u> detail—the objects, their properties, and their interactions—of the real-world system must be dealt with in one way or another:

1. Explicitly represent it,
2. Account for it without explicitly representing it, or
3. Ignore it.

Trade-offs must always be faced. Explicitly represented details increase the complexity of the model and its input data. Implicit details increase the difficulty of configuring and understanding the model and its input data. And ignored details decrease either the accuracy or the flexibility of the model. The abstraction process must work like a sieve, allowing insignificant details to fall through while holding on to the essential ones.

Summary

Land combat models are generally large and complex, due partly to the nature of land combat itself and partly to the broad range of study questions each model of land combat must address. The size and complexity may also be partly due to design decisions and development methods that are actually detrimental to the validity and usefulness of the model.

At the very beginning of the design process, a designer must take utmost care in identifying the essential objects, attributes, and relationships of the modeled system. The designer cannot rely on computer power to avoid the problem of having to decide which elements are significant. Each new detail—a new object, a new attribute, or a new relationship—adds to the complexity of the model, making it harder to understand, increasing its input data requirements, and obscuring its fundamental structures. On the other hand, the designer must be careful not to omit any significant details. This type of error has a much larger effect on the validity of the model than including insignificant details.

With the static structures of the model in place, the model designer next addresses the system's dynamics. For robustness and flexibility, the simulation's decision mechanism must be implemented in a manner that provides the end user with maximum control. Imbedding a decision process in code not only makes the model too rigid but increases the likelihood that the code will have to be changed, thereby endangering the integrity of fundamental model structures.

For the entire life cycle of a combat simulation, careful configuration control and documentation maintenance must be the rule. Model builders and users alike must recognize from the outset that any given model will be expanded and refined many times in the course of its life. If validity is to be maintained rather than destroyed in this process, those who change the code or alter the data must do so with a full understanding of the consequences of their actions.

3 THE ROLE OF THE ENGINEER

Introduction

To design a model of the engineer function for a land combat simulation, the designer must start with a thorough understanding of the engineer mission both for the U.S. Army and for possible threat forces. The designer must understand what engineers are required to do, how they accomplish their tasks, and how their actions support the warfighting capabilities of the force as a whole. Designers must strive to understand the underlying cause and effect relationships and to establish the range of alternative possibilities that will define the boundaries of the model's representation of system objects and system dynamics.

The Combat Engineer in the U.S. Army

Engineer Functional Areas

The five U.S. Army engineer functional areas are: mobility, countermobility, survivability, general engineering (sustainment engineering), and topographic engineering. The primary documents describing engineer operations in general and their respective functional areas in detail are:

FM 5-100, *Engineer Combat Operations*
FM 5-101, *Mobility*
FM 5-102, *Countermobility*
FM 5-103, *Survivability*
FM 5-104, *General Engineering*
FM 5-105, *Topographic Operations*

These six documents provide a wealth of information about combat and construction engineering functions. If the reader is not familiar with these documents, the author recommends a quick scanning of them at this point before proceeding. Such an initial exposure to these documents will provide an overview of the real-world systems the author is attempting to simulate. A more detailed study of their contents yields the answers to many of the design questions that arise in actually building a model of combat engineers.

Lesson Learned: The six basic engineer field manuals (FM 5-100, FM 5-101, FM 5-102, FM 5-103, FM 5-104, and FM 5-105) contain most of the fundamental rules required to es.ablish an expert system for combat engineers as they function under AirLand Battle doctrine.

Any description of engineers and their role in combat would be based on these six field manuals and would repeat the manuals' contents. What follows, then, is not an attempt to provide a complete description of the combat engineer function, but an attempt to call from these field manuals some of the fundamental concepts that will affect the design of the model in the next chapter.

The following pages contain a general description, the goals, and some of the guiding principles connected with each of the five engineer functional areas.

MOBILITY

General Description

The mobility function includes all efforts required to allow the fighting force to move at will. Engineer terrain analysis and reconnaissance identifies the best routes for movement, and engineers assigned to the lead elements provide rapid, in-stride breaching of obstacles. Obstacles may be natural (e.g., rivers), cultural (e.g., embankments), or reinforcing (e.g., minefields and antitank ditches). The mobility function also includes construction of combat trails through areas where routes do not exist and forward aviation combat engineering (FACE) involving the expedient development and repair of landing strips, low altitude parachute extraction systems, and forward arming and refueling points (FARPs).

Goals

To sustain the momentum necessary to retain initiative. [p 1-10, FM 5-101]

To overcome obstacles in stride by planning, forward deployment of equipment, and standardized execution. [p 1-11, FM 5-101]

To allow a force to move rapidly, mass, disperse, and resupply. [p 1-10, FM 5-101]

To provide avenues of approach unexpected by the enemy because of difficult terrain. [p 46, FM 5-100]

To provide early detection of obstacles to movement. [p 2-10, FM 5-101]

Guiding Principles

Bypass obstacles if possible and breach only if no alternative exists. [p 4-7, FM 5-101]

Prepare for overcoming obstacles and performing gap crossings as a part of the maneuver commander's plan, requesting required engineer resources early in the process. [p 3-3, FM 5-101]

Locate engineer countermine/counterobstacle equipment well forward in the leading units to assist with mobility tasks. [p 43, FM 5-100]

Locate countermine equipment (plows, rollers) organic to maneuver units with the lead elements. [p 4-8, FM 5-101]

Execute under cover of darkness or smoke if possible to reduce vulnerability. [p 1-11, FM 5-101]

Execute under small arms fire after suppression of tank, antitank guided missile, and artillery fire. [p 1-10, FM 5-101]

Use electronic warfare (EW) to reduce enemy detection of intentions. [p 1-11, FM 5-101]

COUNTERMOBILITY

General Description

The countermobility function includes all efforts aimed at restricting enemy movement and preventing execution of the enemy. Engineers emplace obstacles to reduce the enemy's ability to maneuver, to increase his* vulnerability to direct and indirect fire, and to protect friendly forces from counterattack. Tactical obstacles include minefields, destroyed bridges, antitank ditches, wire entanglements, abatis, etc. Such obstacles may be used individually or as components of an integrated obstacle system.

Goals

To "delay the enemy's advance, upset his timing, disrupt and channelize his formations, and delay or destroy follow-on echelons." [p 37, FM 5-102]

To hold enemy forces within range of friendly weapon systems to enhance the effectiveness of their fires. [p 38, FM 5-102]

To allow defense of a selected area with an economy of force. [p. 37, FM 5-102]

To protect flanks in the offense. [p 38, FM 5-102]

Guiding Principles

Control obstacle zones centrally to ensure obstacle plan is integrated with and supports the tactical plan. [p 44, FM 5-102]

Execute obstacles at the time and place desired to avoid hindering friendly movement. [p 48, FM 5-100]

Use obstacles "to develop engagement areas in which enemy maneuver is restricted and slowed, thereby increasing the hit probability of friendly direct and indirect fire." [p 38, FM 5-102]

Design obstacles according to the tactical requirements for disrupting, turning, fixing, or blocking the enemy. [p 55, FM 5-100]

Integrate obstacles with existing obstacles and with other reinforcing obstacles. [p 38, FM 5-102]

Cover obstacles with defending fires, targeting enemy breaching equipment to ensure maximum obstacle effectiveness. [p. 58, FM 100]

Make obstacles no more difficult and no less difficult to breach than to bypass. [p 40, FM 5-102]

Design obstacle systems in depth to force the repeated deployment of enemy counterobstacle forces, thereby slowing enemy movement, wearing down enemy resolve, and exposing counterobstacle equipment to loss. [p 41, FM 5-102]

Employ the element of surprise to increase obstacle effectiveness. [p 41, FM 5-102]

*The masculine pronoun is used without respect to gender.

Hold some obstacle emplacement capability in reserve to be used to develop situational obstacles [p 58, FM 5-100]

SURVIVABILITY

General Description

The engineer survivability function includes all efforts at protecting personnel, weapons, and supplies from exposure to both direct and indirect fire. The effort focuses primarily on constructing protective positions for combat vehicles, direct fire weapons, artillery and air defense systems, command and control elements, and critical logistics assets. This function also includes the proper use of camouflage and deception to conceal the location of key force components.

Goals

To limit personnel and equipment losses by reducing exposure to acquisition, targeting, and engagement. [p 58, FM 5-100]

To increase effectiveness of weapons in fighting positions. [p 1-1, FM 5-103]

Guiding Principles

Pass the responsibility for developing positions for individual and dismounted crew-served weapons to each maneuver unit, while available engineers assist with major survivability tasks beyond unit capability. [p 1-7, FM 5-103]

Concentrate engineer survivability support on missions that require unique engineer skills or equipment. [p 1-2, FM 5-103]

Enhance existing terrain through continual improvement. [p 1-6, FM 5-103]

Prepare protective positions for artillery, air defense, command and control, and logistics in the offense and in the defense. [p 1-7, FM 5-103]

Protect facilities emitting a strong electromagnetic signal, or substantial thermal or visual signature. [p 1-7, FM 5-103]

GENERAL (SUSTAINMENT) ENGINEERING

General Description

The engineer sustainment function includes all efforts required to sustain the fighting force. This includes replacing assault and tactical bridging with fixed bridging, clearing obstacles, maintaining and improving lines of communication, and constructing and repairing required facilities.

Goal

To support theater armies with both vertical and horizontal construction, maintenance, and repair.

Guiding Principles

Give first priority effort to direct support of military forces, including repair to damaged air bases, other critical facilities, and lines of communication (LOCs). [p 5, FM 5-104]

Expect greater demands for sustainment engineering during offensive combat because LOCs lengthen, new needs for logistics facilities arise, and obstacles breached or bypassed by leading elements must be fully cleared. [p 44, FM 5-100]

Make maximum use of existing facilities and host nation capabilities and resources. [p 3, FM 5-104]

Use austere design and construction techniques. [p 3, FM 5-104]

Plan for phased construction to allow the use of facilities before work is completed. [p 4, FM 5-104]

TOPOGRAPHIC ENGINEERING

General Description

"Topographic engineering defines and delineates the terrain for planning and operations, and provides precise location data to modem efficient weapons systems." [p. iii, FM 5-100] The topographic engineering function includes terrain analysis, production of updated maps and overlays, and survey support for artillery and missile targeting requirements.

Goal

To ensure that timely, accurate, and sufficient knowledge of the battlefield terrain is provided to each commander throughout all phases of combat operations. [p 1-1, FM 5-105]

Guiding Principles

Provide support most often in the form of information. [p 1-1, FM 5-105]

Establish geodetic survey control for precise positioning of artillery and command, control, communications, and intelligence (C^3I). [p 1-7, FM 5-105]

Use standard topographic products provided by Defense Mapping Agency or by host nation through international agreements. [p 1-7, FM 5-105]

Supplement and enhance available data to provide information on the current state of the ground. [p 1-11, FM 5-105]

AirLand Battle Doctrine

The Army's AirLand Battle doctrine describes four basic tenets of how the army fights: initiative, agility, depth, and synchronization. The implications of these principles for engineers are explored in FM 5-100 [p 7].

Initiative covers both the universal concept of a force taking the actions necessary to gain control over when, where, and how to meet the enemy and the narrower concept of allowing subordinates to develop the detailed plans for accomplishing those actions within the broad framework of the commander's intent. Engineers have a parallel twofold requirement under initiative. First, "engineers give maneuver commanders options, not otherwise available, that aid them to be bold and audacious by minimizing their risks and enhancing the mobility of their forces." [p 6, FM 5-100] Second, engineers must understand the commander's intent and anticipate engineering requirements. Because timing is crucial, engineers "initiate preparatory actions before their need is often perceived in detail at higher echelons." [p 7, FM 5-100]

Agility means having the ability to act faster than the enemy. "Engineers are task organized to ensure rapid response to changing requirements. They shift support for the main effort with the minimum delay and with least possible reconfiguration and coordination. They are self-aligning, sustainable, and responsive to maneuver commanders at all echelons." [p 7, FM 5-100]

Depth means using the full range of friendly resources to attack the full range of enemy resources. Engineers must provide support throughout the entire theater of operations, in the offense and in the defense, from the forward combat zone to the rear.

Synchronization means having the ability to bring friendly forces together in the right place at the right time to do the right things. "Engineers plan their activities carefully so their effects are felt at the decisive time and place and in the desired manner. Engineer commanders ensure that the multiplicity of engineer activities spread across the battlefield have a unity of purpose with the rest of the force." [p 7, FM 5-100]

Engineer Organizational Structure

The second chapter of FM 5-100 outlines the typical engineer organizational structure for today's Army but introduces the discussion on engineer organizational principles by stating:

> Strategic objectives, the nature of the TO, and the forces available all influence the design of the theater commander's campaign plan. The requirements for engineer forces and type organizations, which come from this plan and drive the engineer architecture, therefore vary from one theater to another. [p 14, FM 5-100]

This flexibility in the current engineer organization and the promise of major Army restructuring during the 1990's will force the model design to accommodate a wide variety of possibilities. Many of the questions that models will answer in the near future will focus on alternative force structures and the advantages/disadvantages of each. So rather than reproduce the descriptions of the standard engineer structure at each echelon as they appear in the second chapter of FM 5-100, some of the more stable organizational principles that guide the building of an engineer force are cited below:

Task organize engineer forces to requirements.
Give priority to the main effort.
Integrate engineers with maneuver and fire.
Ensure current engineer operations promote future force operations.
Do not hold engineers in reserve.
Build a logistically sustainable force.
Maintain effective command.
Use all local resources. [p 14, FM 5-100]

> As engineers must physically travel to where they are needed, their ability to react quickly to changes in the situation is dependent upon forces being wisely positioned before the battle. There normally is not time to radically restructure the engineer organization for combat or to move engineers across the battlefield after the battle is joined. [p 43, FM 5-100]

Engineers may have several different command and support relationships with maneuver units. "Command authority over engineer units is given to a maneuver commander when that commander requires immediately responsive engineer forces...The command relationship can be attachment, operational control (OPCON), or operational command (OPCOM)." [p 30, FM 5-100] Engineers in a **direct support** relationship provide support to a specified unit while engineers in **general support** provide support in a specified area. In a support relationship, "the engineer unit commander organizes the unit and suballocates tasks in the manner he determines will most effectively meet the needs of the maneuver commander." [p 30, FM 5-100]

> The decision whether to provide engineers in a command or a support relationship to a subordinate maneuver headquarters, then, is a balance between the needs of the higher commander for flexibility and for the most efficient use of scarce engineer assets and the needs of the subordinate commanders for responsiveness and for the ability to task organize his forces...
>
> Normally, the corps commander provides each committed division with a corps combat engineer battalion in a command relationship. Additional corps engineers provided to the division are usually in a support relationship. This gives the division the capability to task organize and provide adequate engineer support to its committed brigades, while additional engineers accept missions in the division rear. [p 31, FM 5-100]

The maneuver commander generally sets engineer work priorities, both between the three main functional areas (mobility, countermobility, survivability) and within those functional areas. [p 3-17, FM 5-101] If the combat posture changes from offensive to defensive, the priority of engineer support must be able to shift rapidly from mobility to survivability and countermobility. If the force is successful and changes back to an offensive posture, then engineers must respond quickly. [p 51, FM 5-100]

Engineers and Combat Service Support

Engineers place two requirements on the sustainment system: support for their basic unit needs and supply of the materials and transportation needed to accomplish their mission.

> A maneuver force that assigns tasks to an engineer unit also has responsibility to obtain the resources needed to perform those tasks...
>
> The materials and transportation needed for engineer missions often compete with the requirements of other units. The maneuver staff satisfies competing demands based upon the commander's priorities. The time-intensive nature of engineer operations must be a key consideration for the staff in resolving those demands. For example, assigning corps ammunition trucks to haul mines initially enables the engineers to immediately start on the obstacle plan. Yet it does not prevent those same trucks from subsequently hauling artillery ammunition needed later for the fire support plan. [p 39, FM 5-100]

Threat Combat Engineers

Much of what has been said above about combat engineers in the U.S. Army is applicable to the engineer organizations of many of our possible threats, the most likely of which adopt Soviet doctrine. Threat combat engineers are responsible for permitting rapid movement of their maneuver forces, for protecting the flanks of assault forces with rapidly emplaced obstacles, and for strengthening a defense with prepared positions and obstacles. Most threat forces seem to adopt at least three of the tenets of AirLand Battle doctrine: agility, depth, and synchronization. Soviet doctrine does not embrace initiative within subordinate elements, preferring instead to control a highly synchronized force at a high command level. Threat doctrine dictates that engineer activities are integrated with the overall tactical plan and must usually be conducted under fire or in advance of the main assault force.

In combat itself, achieving rapid movement and applying constant pressure to overwhelm the enemy take precedence over preserving combat effectiveness, with a depth of reserves to balance the loss of life and equipment. This is an important difference for engineers. Unlike their U.S. counterparts who begin mobility tasks after suppression of all but small arms fire, threat engineers are expected to work under fire, clearing avenues of approach as much as a half day ahead of the assault forces.

Organizationally, Soviet engineer elements are organic to all tank and motorized rifle units down through the regimental level, and significant engineer reserves exist at higher echelons. This reserve element combined with centralized control means that Soviet engineers can rapidly switch their efforts from one area to another, concentrating efforts where they are needed most. As with U.S. forces, many engineer tasks are performed by soldiers of other combat arms. Minefield breaching/clearing may be done by individual soldiers or by tank-mounted plows and rollers of the armored forces. [p 2-3, FM 5-102]

4 REPRESENTING THE ENGINEER FUNCTION IN LAND COMBAT MODELS

Objective

Thus far this discussion of the design requirements for modeling the combat engineer function has not been limited to a particular class of combat models. The concept of a combat model itself contains the implicit assumption that the model represents a horizontal slice of the battlefield with units that move around on a section of land and interact with other units, usually by fighting with or against each other. Beyond this simple element of commonality, the universe of combat models may be split along a number of lines, among them:

- level of resolution (high, medium, low),
- interactive vs. noninteractive,
- analytic vs. training,
- stochastic vs. deterministic, and
- full-spectrum vs. functional-area specific.

USACERL's experience, especially with VIC, makes it easier for the author to discuss a design for deterministic, analytic, noninteractive, midlevel models; and the presentation of the objects, relationships, events, and processes in this chapter will be from that perspective. However, since the intention is to address primarily the first level of abstraction, that of identifying the essential elements and designing a structure to properly represent their interactions, much of what is presented here should apply to any model of the combat engineer function.

In addition to the variations in design resulting from the different classifications listed above, one must also address the variations that result from the different purposes for incorporating an engineer representation into a given combat model. Two distinct views are evident. The first, and more general, is that the combat engineer function is so intertwined with the way units move, fight, and sustain their operations that land combat itself cannot be modeled properly without including at least a minimal representation of engineers. The second, at the other end of the spectrum, is that the need exists for detailed models of the engineer function within the framework of a combat environment to answer engineer-specific questions about force structure, systems acquisition, and tactics. The level of detail at which engineers are played in these two extremes may vary from an implicit engineer presence with aggregated computation of capabilities to an explicit simulation of engineer activities fully integrated with the model's other battlefield functions and sensitive to changes in the factors being studied.

Regardless of how much engineer detail is explicitly included in the simulation, the analyst must ultimately account for all of the essential details of the engineer processes if study results are to be valid. If the code of the model is not accounting for an essential detail during a run of the simulation, then off-line calculations must be made to ensure the reasonableness of scenario assumptions. This approach adds complexity to the analytic process and opens the possibility for overlooking important factors or miscalculating their influence. Ideally, the simulation design should offer the flexibility of allowing the user to vary the level of engineer detail being automatically considered during the simulation in much the same way that a physical scientist controls the influence of factors in a laboratory experiment.

The goal is to improve battle simulations by designing a structure for the combat engineer function that will portray both what engineers do and the effect of what they do in a natural and flexible framework that allows the model user to control the level of detail. If the design is done well, the resulting model will satisfy the span of analytic requirements. To do this, one must not lose sight of the fact that the resulting model is a tool for analysis and that the internal structures of the model must provide a shell of natural behaviors to drive the flow of events that spring from the controlled behaviors dictated by the model user. This calls for a high level of abstraction, inclusion of only the essential details, and a faithful representation of the processes and relationships that define the dynamic behaviors of the real-world system.

The Essential Elements of an Engineer Representation

FM 5-100 provides a first step for the move from real world to model, especially in the diagram reproduced in Figure 1. The activities that result from the interactions of the maneuver force commander with his engineer support can be placed into the following several categories for modeling.

1. Those that are obviously representable as a process or behavior within the internal model structure:

- provide friendly engineer capabilities,
- integrate logistics, and
- decide engineer tasks and priorities.

2. Those that require such a high-level of information, decisionmaking, and control that they are typically handled externally and entered as scenario data:

- provide terrain analysis,
- recommend routes,
- recommend obstacles,
- recommend fortifications,
- decide commander's concept, and
- decide engineer tasks and priorities.

3. Those that might be represented as an internal process or behavior but that have been difficult to implement:

- recommend routes,
- recommend obstacles,
- recommend fortifications,
- decide engineer tasks and priorities, and
- integrate obstacle concept with maneuver and fires.

The difficulty in implementing the items in the third category arises both from the fact that current technology in the area of expert systems and artificial intelligence cannot easily handle the complex decision process involved with these activities and from the fact that the resolution of a given model may be such that the required information for the decision process is not available to the internal structures. For example, route selection and obstacle planning in a Corps-level battle simulation require more terrain data than provided by the typical 3- to 4-km grid size.

With the acknowledgement that the third category is beyond current experience, this report concentrates on establishing a sound structure for the other two categories, primarily focusing on representing engineer capabilities given that the scenario data will define the commander's concept and identify the engineer tasks and priorities needed to support it.

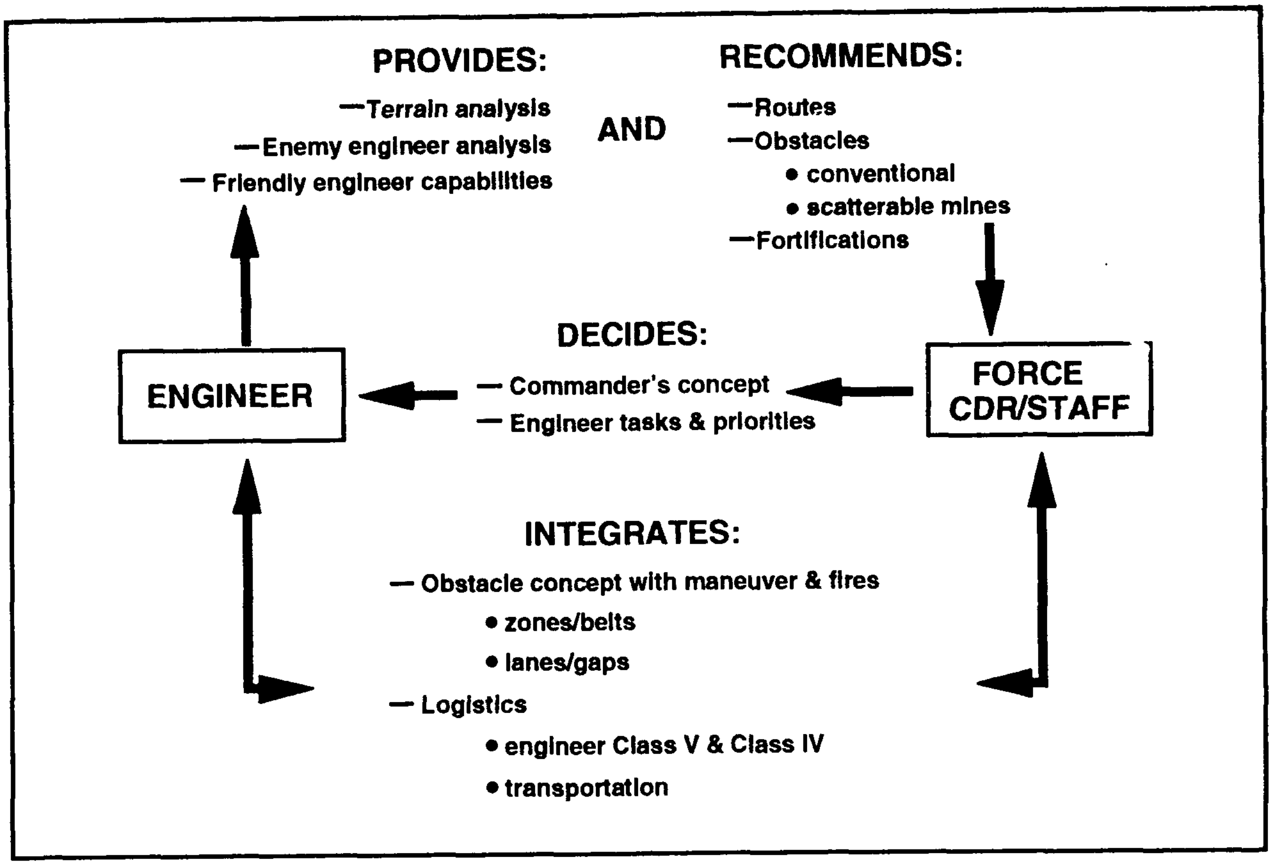


Figure 1. Interactions Between the Maneuver Force and Its Engineer Support (from FM 5-100, p 26).

Engineer Tasks

The starting point for modeling combat engineers is to look at what they do: the tasks they perform under one of their main mission areas (mobility, countermobility, survivability, and sustainment). As a part of their work in developing the EMIP plan, the Engineer Studies Center conducted an expert judgment survey to identify the specific tasks from each engineer mission area that are most important to the success of the combined arms team; these tasks (Figure 2) are the foundation of the combat engineer model. Each task representation has two relatively independent components: the engineer **effort** involved in doing the task and the **effect** that the performance of the task has on the way other units move, fight, and sustain their operations.

In the diagram of Figure 3 showing the interface between a combat model and the engineer module within it, the representations of the two components–effort and effect—of each engineer task manifest their relative independence by occurring in different parts of the model. The engineer effort involved in task performance is represented in the engineer module, which is responsible for what the model sees of engineer units, resources, methods, and missions, and for the processes that link those crucial elements together to determine what engineers do and when they do it. The engineer effect of task performance is represented in the way other units accomplish their missions, which is handled in parts of the model outside of the engineer module. The current approach is to consider the two components as separate issues, both because of the independence of their representations and because of the distinctly different structures they require.

MOST IMPORTANT ENGINEER TASKS TO INCLUDE IN COMBAT MODELS

MOBILITY

Breach obstacles in the assault (breach minefields, span short gaps,...)
Improve assault breaches for follow-on forces (clear minefields, widen lanes,...)
Conduct river crossing operations in the assault
Improve river crossing sites for follow-on forces
Prepare and maintain combat trails/roads
Prepare and maintain forward airlanding facilities

COUNTERMOBILITY

Install linear obstacles (minefields, tank ditches,...)
Install point obstacles (road craters, bridge demolition,...)

SURVIVABILITY

Prepare fighting positions for direct fire systems (tanks, TOWS,...)
Prepare positions for indirect fire & other systems (artillery, ADA, CP,...)

SUSTAINMENT

Maintain main supply routes
Repair airfield damage
Prepare and maintain sites for CS and CSS units
Prepare and maintain rear area facilities
Construct and repair port and waterfront facilities

Figure 2. Engineer Mission Area Tasks Important to the Success of the Combined Arms Team.

The Engineer Effect

Regardless of the resolution of a combat simulation, the representation of the effect of engineer activities is crucial to the simulation's realistic portrayal of a modern battlefield. On that battlefield, river crossings significantly affect units of all echelons and types; obstacle belts and zones limit an enemy's mobility; fighting from prepared positions enhances firepower and survivability; damaged roads hamper movement of both combat and support units; and damaged runways can severely curtail air support.

Although the author will later be able to make a general proposal for a basic structure to represent engineer effort, no such proposal can be made here about representing the engineer effect. Modeling the effect of engineer activity is both model-specific and task-specific. It is an integral part of the basic design of a combat simulation. And it is the most important but perhaps most difficult element of an engineer representation. The purpose of specifically listing the goals of each mission area in Chapter 3 was to establish a basis for what the representation of the engineer effect must attempt to capture. The combat model designer must begin with an understanding of those basic goals if the resulting model is to provide any measure of the contribution of engineers to the combined arms team.

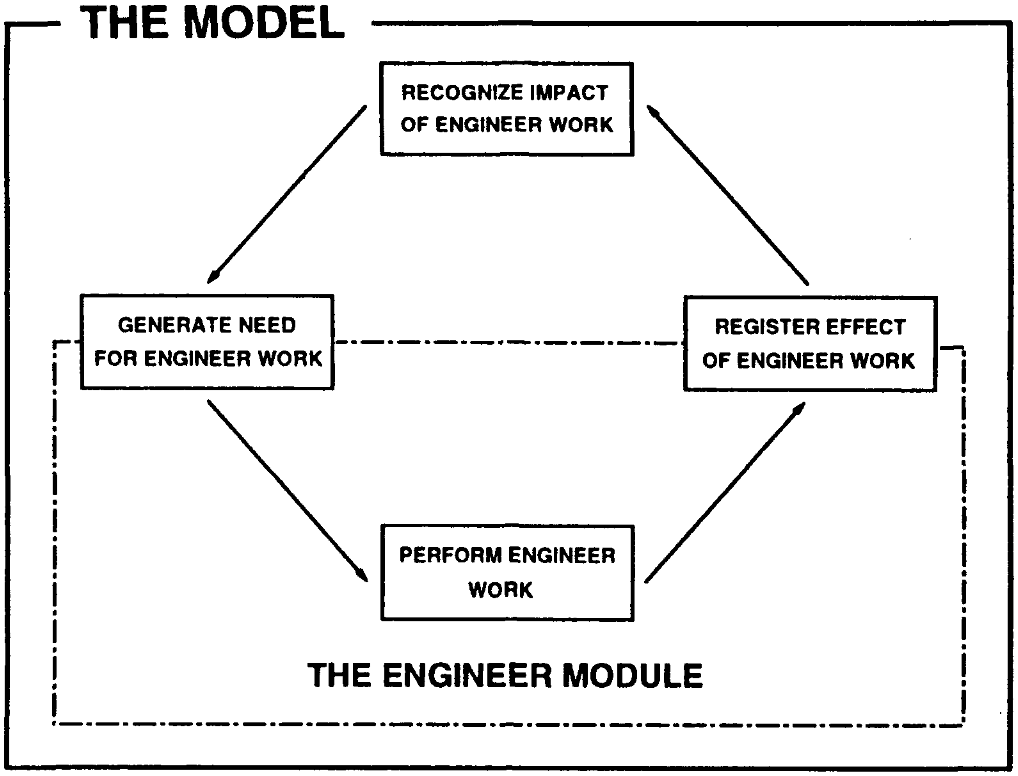

THE MODEL
RECOGNIZE IMPACT OF ENGINEER WORK
GENERATE NEED FOR ENGINEER WORK
REGISTER EFFECT OF ENGINEER WORK
PERFORM ENGINEER WORK
THE ENGINEER MODULE

Figure 3. Conceptual View of the Combat Model and the Engineer Module.

USACERL's modeling experience in the area of engineer effect is not as extensive as their experience with representing the engineer processes of task performance because their efforts have usually been devoted to enhancing existing models rather than designing new ones. Being thus presented with established structures for unit movement and engagement, the researchers backed into the next lesson from the direction of having often been unable to extend certain parts of an engineer representation because unchangeable structures prevented further development.

Lesson Learned: The most crucial time during the life cycle of a combat model for adding engineers is at the very beginning of the design process. If the basic movement and combat algorithms are designed without the engineer function in mind, mistakes will surely be made that will not be correctable after the model is in production use.

This lesson was particularly applicable to the work with VIC. For example, its ground unit movement is based on paths scripted in the scenario and unalterable during a run of the simulation. In addition, it represents unit movement and engagement as two relatively independent activities. The combination of these basic design elements precludes integrating an engineer countermobility plan with the defensive tactics of the maneuver force. The contribution of the engineer obstacles is reduced to a minor delay and a small amount of attrition, with no synergistic enhancement of covering fires or canalization of enemy forces. Even worse for the model as a whole, the effects of all four AirLand Battle tenets are minimized in its simplified representation of how units move and fight; agility, initiative, depth, and synchronization play only a small part in determining the outcome of an engagement.

Properly determining the outcome is at the core of why one uses battle simulations for analysis. If the simulation fails to portray what would really happen, any effort at measuring the influence of contributing factors is useless. It follows that furious activity on the part of the simulated engineer force has little value if the proper effect of that activity is lost.

Lesson Learned: Modeling the effect of engineer task performance is more important than modeling the task performance itself.

Lesson Learned: The model cannot measure requirements on engineer effort with any accuracy at all in the absence of a commensurate representation of the effect of the engineer work.

Lesson Learned: If the effect of an engineer task cannot be modeled, the performance of the task should not be modeled because the level and priority of engineer effort are generally functions of the effect and cannot be calculated realistically in the absence of an effect.

Representing the effect of engineer activity is so important to the analysis of the flow of events in a combat simulation that provisions must be made for the analyst to dictate desired effects in a scenario without regard for the engineer effort required to achieve them.

Lesson Learned: The user should be able to model the effects of a functional area without having to simulate the implementing process (i.e., the user should be able to make simplifying assumptions about the way the simulated battle is to evolve).

To this end, the design of the model and its scenario data should allow reinforcing obstacles to appear at a specified location at a specified time without engineer effort, allow maneuver units to move through natural and reinforcing obstacles at a predetermined speed without consideration of the engineer breaching capability at hand, allow units to fight from designated prepared positions without representing the effort to prepare them, and allow support facilities and lines of communication to function with no possibility of damage.

Using the option of modeling the effect of engineer task performance without accounting for the corresponding engineer effort requires caution and understanding on the part of the scenario developer. Since it is important to build realistic battle simulations, the first rule for using such simulations should be to avoid unrealistic scenarios. In other words, units moving across the simulated battlefield should move at appropriate speeds whether or not engineers are visibly present to help them, and the terrain should include only as many manmade obstacles and positions as could reasonably be emplaced. The difficulty with modeling these types of effects without the corresponding engineer effort to achieve them is that the scenario becomes insensitive to changes in engineer capabilities and many of the compensations in data that must be made to preserve realism are difficult to compute off-line. So even though the design should be flexible enough to offer the possibility of simulating effect without effort, it must also include the other extreme of simulating effect only with effort.

If a combat model is to portray the full engineer process from effort to effect, then the nonengineer module structures must support the following minimal requirements:

- Units must be able to recognize the presence of the physical feature altered by an engineer task and to react appropriately to it.

- Each unit must suffer delays and attrition from encountered obstacles according to that unit's ability to breach or bypass; and each type of obstacle must have its own distinct algorithms for computing encounters, tactics, delays, attrition, and future effectiveness.

- Units must use prepared positions properly and have appropriate attrition from direct and indirect fire.

- Combat trails must speed unit movement through difficult terrain.

- Obstacle breaches must decrease the delay and attrition of an obstacle.

- The condition of a road must affect the speed of units using it.

- The condition of runways must affect air sorties.

- The presence and condition of port facilities, depots, and lines of communication must affect the flow of supplies.

To achieve an ideal representation of effect, the model must be able to recognize and account for the synergism that arises when several interdependent tactical situations occur simultaneously. For example, a defensive force firing from prepared positions on an enemy crossing a minefield will produce larger enemy attrition than the sum of attrition assessed from each situation considered independently. Such a capability would not only produce a more realistic assessment of the engineer contribution but also a more realistic battle simulation as a whole.

The Engineer Effort

If you think of a full-spectrum battle simulation as providing a miniature world with which to experiment, then the requirements for such a model's representation of combat engineers and their activities could be phrased as a description of what you would expect to see:

- Engineers perform the types of tasks normally required of them.

- Engineers respond correctly to the evolving situation.

- Engineers follow the command and control structures of the scenario.

- Engineers have realistic performance levels with a proper assessment of the effect of resource availability and situational factors (weather, visibility, level of combat, work load, terrain).

- Engineers suffer appropriate levels of attrition from direct and indirect fire.

- Engineers move and work with appropriate effects from terrain and obstacles.

- Engineers have proper interactions with other combat, combat support, and combat service support elements.

- Nonengineer units perform engineer functions commensurate with their ability.

The question must be asked as to whether a combat model must actually simulate engineers and all of their activities in order to meet these requirements. Can the model be designed to accomplish the substance of the above description without actually simulating engineers and their activities? What was said in Chapter 2 about the complexity of nature overpowering the computer and the analyst is true; no one can replicate all of the detail of the real world regardless of the massive computer power available. Even if the choice is to simulate the engineer activities, decisions must be made about the level of detail to which that simulation can go and still maintain validity. This is the point at which appropriate abstraction and simplification is required.

Stated simply, the model will properly portray combat engineers and their activities if the designed engineer structure uses the typical scenario descriptions of units, terrain, resources, and missions as a basis for assessing engineer capability so that the mission accomplishment during the simulation is sensitive to the very factors that affect real-world engineers:

- the number and type of resources available,
- the organization and location of units and equipment,
- capabilities of equipment,
- availability of supplies,
- various methods available for accomplishing each type of task, and
- situational conditions:
  - unit work load,
  - advanced planning,
  - synchronization requirements,
  - enemy activity,
  - friendly activity,
  - terrain,
  - visibility, and
  - weather.

To be more precise, one would like the model to be able to determine if and when any particular engineer task will be accomplished and to include all of the above factors appropriately in the process.

Since many of the factors listed above are situational, the immediate inclination is to simulate the engineer task performance as an integral part of the battle simulation in order to account for the influence of each situational factor as it evolves. Simulating engineer activity in detail is appropriate for a high-resolution battle simulation because the resolution is closest to representing real-world entities in a one-to-one mapping. The problem with high-resolution simulations is that they simulate a time span that is too short to see the entire engineer process for most tasks. A combat model's timespan needs to be longer than a day to capture engineer activity, so designers must turn their attention to lower resolution simulations. Doing that immediately raises a resolution mismatch problem, and the need for—and even the validity of—simulating engineer task performance in such models must be examined carefully.

A resolution mismatch arises in fitting an engineer representation into a combat simulation because of the classical modeling tradeoff between time/space and detail. As a combat model's vision expands in time and space, detail must be lost; and as the level of detail expands, the scope of time and space that can be accurately represented shrinks. The combat engineer function is caught in the middle by its requirement for both expanded time/space and expanded detail. The source of the resolution mismatch is the representation of engineers themselves and of the processes they use in accomplishing their mission. This occurs because engineers generally work at the company or platoon level over a period of time exceeding an hour or two on an individual terrain feature or facility. Corps-level models simulate several days of combat but are not generally designed to handle such small units or to be sensitive to the effects of small, isolated features. In theater-level models, even the processes of units larger than a battalion are difficult to capture, so the true effect of an engineer presence may not be detectable.

Some model designers have handled the resolution problem by accounting for the engineer effort without explicitly representing it or by allowing the effect to be modeled with no apparent attempt to account for the effort required. Examples of these approaches are: accounting for engineer bridging and breaching capability by appropriate adjustments to input data for river crossing times and obstacle delays and attrition; making the level of position preparation a function of the amount of time the unit has been at the location; and allowing obstacles like minefields and tank ditches to simply appear on the simulated battlefield with no representation of the effort required to emplace them.

These approaches have several drawbacks. Using "nonengineer" data to account for engineer capability packs too much information into a few input data items, making scenario development more difficult and obscuring the influence of the individual factors that affect capability. Additionally, scenario excursions are insensitive to changes in engineer capability unless those very complicated data items are recalculated; the expert engineer guidance required during scenario development is not always available during excursion runs. Assumptions in the input data about engineer capability fail to take into account crucial situational factors like the level of demand for engineer resources, the possible attrition of key engineer equipment, and the effect of timing and synchronization difficulties. Finally, allowing the model to represent the effect of a task completion without a measure of the effort required to produce that effect can lead to dangerous conclusions during the complex analysis of sorting out what happened and why. These approaches yield an engineer capability that is ever-present, indestructible, and always adequate to meet the demand. Such a representation of engineer capability for a combat model limits the range of the model's study applications and increases the effort required to obtain valid results.

Returning to the problems inherent in explicitly representing an engineer presence in a battle simulation, the resolution mismatch requires a closer look. Two things must be done at this point: (1) identify more precisely why the mismatch occurs to determine what suffers if the combat model actually includes the inconsistency of resolution, and (2) explore alternatives that avoid the inconsistency problems but still explicitly account for engineer capability. Looking again at the diagram of the interface between a combat model and the engineer module (Figure 4), the task is to locate the possible inconsistencies that arise between a high resolution engineer module and the low-resolution combat model containing it.

Lesson Learned: High-resolution detail of the processes of the engineer module remains consistent with a model's low-resolution processes as long as the processes themselves do not mix the low- and high-resolution details.

In the low resolution flow of the battle, events occur that generate missions or tasks for engineers. The most important response from the engineer module is to properly schedule completion of the task. Perhaps the easiest, though imprecise, action would be to consult an input table of work times and simply schedule completion appropriately. The other extreme would be to have scenario data that defines the engineer resources and unit structure to the last shovel and occupational specialty and to use that information in simulating the mission assignment and performance process in detail. Such attention to the detail would hopefully produce a more realistic completion time because that approach takes into account so many more influencing factors. The two areas of concern are (1) that the detailed simulation actually produces a better completion time, and (2) that the nonengineer functions of the model are not improperly affected by the engineer presence being simulated.

For example, when a maneuver unit unexpectedly encounters a destroyed bridge on a river it has to cross, a bridging mission for engineers is generated and the maneuver unit waits. In the first approach, the engineer module's action would be to determine a likely time for the bridge completion and schedule it, triggering the maneuver unit to cross when the work is finished. At the other extreme, a detailed simulation of the bridge building process has the advantage of taking into account the timing variations that result from different bridging techniques, different unit organizations and resource distributions, a range of demands on resources, environmental effects at the work site, etc. As long as the high level of engineer detail is used simply to compute a better completion time, model validity is not endangered. Essentially, the rest of the model does not need to know how the completion time was determined, and the method used can include as much detail as desired if it produces an appropriate answer.

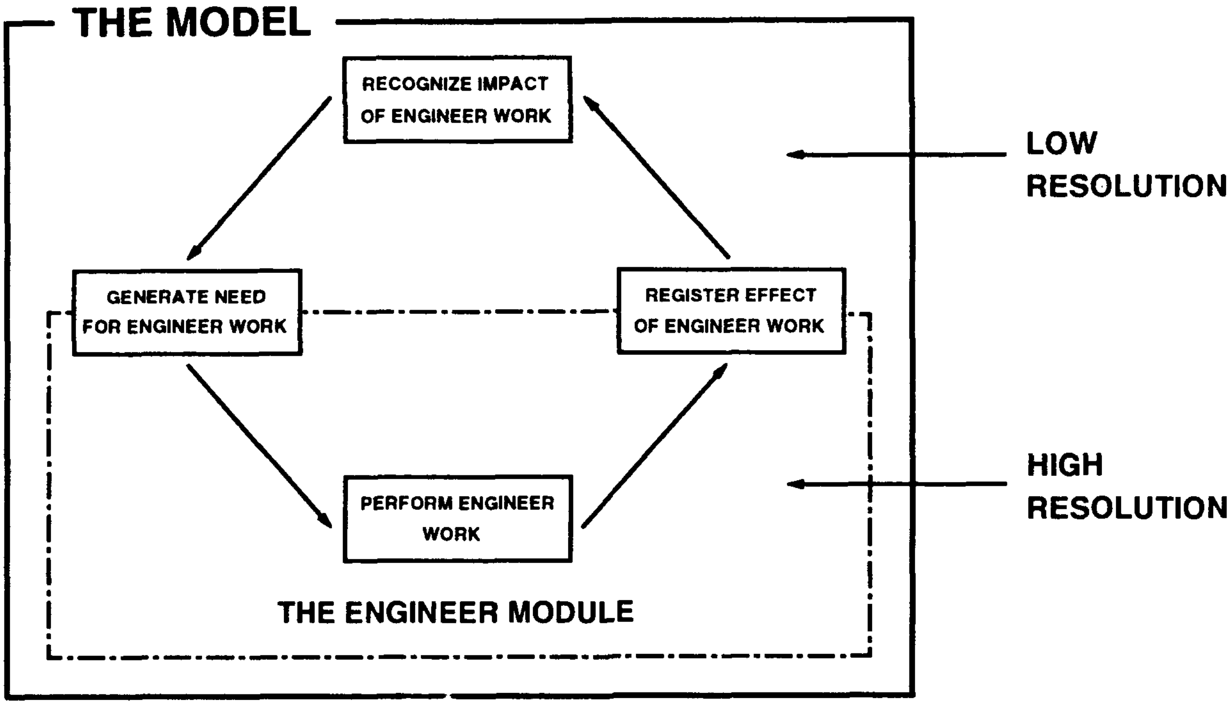

THE MODEL
RECOGNIZE IMPACT OF ENGINEER WORK
GENERATE NEED FOR ENGINEER WORK
REGISTER EFFECT OF ENGINEER WORK
PERFORM ENGINEER WORK
THE ENGINEER MODULE
LOW RESOLUTION
HIGH RESOLUTION

Figure 4. The Resolution Mismatch Problem of Representing Engineer Processes in a Low-Resolution Combat Model.

Where is the resolution mismatch in this? It is not that there is too much detail for engineers, but that some of the engineer processes may be using algorithms and data that may not produce good results for small units. For example, in simulating the movement of an engineer work team from its base camp to the work site, the natural inclination is to use the model's movement algorithms. In that way, all of the terrain factors that affect unit movement in general will be taken into account for engineers as well. But the movement algorithms are designed for much larger units, so the delays from obstacles and the hindrances of ground cover and elevation and any other factors affecting large units may not be very appropriate for small work units. And when such a unit encounters an enemy, the attrition calculations will have a similar problem. Even more detrimental, the algorithms handling the actions of a large unit when it encounters an enemy may be totally inappropriate when the enemy is an engineer platoon.

Lesson Learned: To decide whether detailed simulation of engineer processes maintains model validity, examine each step where the high-resolution engineer process overlaps the low-resolution portion of the model to verify that the result is acceptable.

In the example, if you decide that the presence of small engineer work units causes an inappropriate turn of events, the option of eliminating the simulation of the engineer task performance sequence can be taken without too much loss of detail. That approach might actually arrive at a more accurate estimate of the work unit's travel time by calculating it than by simulating it. This would avoid the many problems that arise in trying to automate an intelligent path selection for engineer work teams but eliminates the possibility of enemy interdiction of the task performance. Calculating the travel time rather than simulating the movement requires also calculating the time to perform the work in order to preserve consistency.

In USACERL's work with VIC, all four of the areas in the conceptual diagram presented resolution mismatch problems.

Perform Engineer Work. The main problem was simulating the movement of a small engineer work unit with a dynamically generated path. In a complicated terrain environment, the implementation will suffer from having an excessive travel time assessed because the work unit's path choice will be less than optimal to portray realistic movement.

Generate Need for Engineer Work. The researchers were unable to implement a mechanism for dynamically generating new obstacle emplacements in response to the evolving situation. The terrain and intelligence information required to automate the engineer decision process was simply not available in VIC. Ultimately, control of the countermobility plan was left to the scenario data, which was enhanced to allow the user to manage obstacle emplacement by specifying a window of opportunity for timing the emplacement activity.

Register the Effect of Engineer Work. The major difficulty in this area resulted from the mismatch of maneuver unit size to obstacle size. Obstacles such as minefields and tank ditches have little delay effect on a unit the size of a battalion but do cause a commitment of engineer breaching resources for a substantial amount of time. The solution was to allow the maneuver unit to continue moving before the breach was completed but to leave the breaching equipment behind to finish the task. This separated the breaching effort from the effect of the breach but produced valid unit movement and use of resources.

Recognize Impact of Engineer Work. This area also suffered from the mismatch of maneuver unit size to obstacle size. VIC's relatively small obstacles could not be integrated with a plan for covering fire. In a joint effort with a team from the U.S. Army Waterways Experiment Station, Vicksburg, MS, the researchers implemented a new obstacle type representing what would be accepted doctrinally as an obstacle zone or belt. This new obstacle type was more closely aligned in resolution with the maneuver units encountering them and made synchronization with covering fire possible.

One of the most important decisions regarding the representation of engineer effort and the processes involved with task performance in a Corps- or theater-level battle simulation was:

> **Lesson Learned: The primary reason for explicitly modeling engineer task performance is to improve the accuracy of estimates of engineer capability, i.e. what engineer tasks can be done and when. The battlefield influence of engineers is in the timing and the effect of their task completion and not in their presence per se or in the simulated performance of their tasks.**

Such an approach helps to narrow the resolution mismatch problem if researchers wish to represent the engineer processes in detail. This lesson may not be completely true for high-resolution battle simulations that represent the effects of an engineer presence apart from task accomplishment (e.g., intelligence about enemy engineer activity being used to predict enemy plans, specific targeting of engineer equipment, the effectiveness of engineer reconnaissance). But lower resolution models usually do not have the detail to capture such things.

With the decision that the timing of engineer task completion is the most important part of modeling engineer capability, a breakdown of factors that affect the overall time will be useful.

Time mission is complete =

Clock time when mission generated +

Time to synchronize with other elements +

Time to assign mission to unit +

Time to form work team +

Time to prepare equipment and load supplies at base +

Time to move equipment and supplies to site +

Time to work at site.

Each of the summands is affected by some of the factors identified at the start of this section, and some of those factors are affected still further by others:

Clock time when mission generated is affected by:

- Effect of reconnaissance/intelligence,
- Effect of planning,
  - Prior planning for known missions,
  - Contingency planning for expected missions,
- Situationally-controlled mission generation,
  - Response to enemy action, and
  - Response to friendly action.

Time to synchronize with other elements is affected by:

- Suppression of enemy fire,
- Smoke cover,
- Night cover,
- Enemy activity, and
- Friendly activity.

Time to assign mission to unit is affected by:

- Engineer unit organization and placement, and
- Command and control requirements.

Time to form work team is affected by:

- Equipment availability,
  - Engineer unit organization and placement,
  - Changing job load/work priority,
  - Amount of equipment lost to attrition/failure,
  - Type of technique being used,
    - • Criteria for choosing technique,
      - Speed/effect/most likely,
      - Location of site in relation to FEBA,
    - • Engineer unit organization and placement,
    - • Changing job load/work priority,

- Supply availability, and
- Organizational time.

Time to prepare equipment and load supplies at base is affected by:

- Type of equipment being used,
- Type of technique being used, and
- Size of specific task.

Time to move equipment to site is affected by:

- Distance from base to work site,
  - Engineer unit organization and placement,
  - Method of travel,
    - Land,
    - Airlift,
  - Travel criteria for choosing path,
    - Avoid known obstacles,
    - Avoid exposure/detection,
    - Move with haste,
    - Move on road,
    - Assess impact of intervening terrain,
    - Assess congestion along path,
- Speed of equipment/work team,
  - Method of travel,
  - Type of equipment,
  - Condition of equipment,
  - Terrain being traversed,
  - Congestion along path,
  - Night/weather/visibility conditions,
- Delays from enemy interdiction, and
- Delays from encountered obstacles.

Time to work at site is affected by:

- Type of technique being used,
- Size of specific task,
- Type and amount of equipment present,
  - Amount originally assigned,
  - Amount lost to attrition/failure,
  - Amount lost to preemption,
- Set-up time for equipment,
- Situational factors,
  - Combat/exposure,
  - Day/night,
  - Weather,
  - MOPP level, and
  - Soil conditions.

One conclusion to draw from examining this list is that the effect of many of these factors can be determined from a relatively simple process using input data that has an intuitively natural structure. Another is that extreme measures to determine any one of the contributing times accurately may be unwarranted if other contributing times are likely to be grossly imprecise. Still another is that the question of whether or not to simulate the engineer processes revolves around how to take into account the factors that affect the computation of the last six times (time to assign mission to unit, time to form work team, time to prepare equipment and load supplies at base, time to move equipment and supplies to site, time to work at site). If the overall model structure prevents simulating accurately, then calculated times with some of the factors ignored might give better results. Finally, one sees that the six time factors forming the core of the representation of engineer effort are almost entirely independent of the type of task being performed.

Lesson Learned: The basic processes in representing engineer task performance are the same for all task types and can be modeled by a single sequence of procedures.

Using the same task performance sequence for every type of task greatly simplifies the representation, whether task perfomance is simulated or not. With this approach, the minimal requirements for building a structure that can determine if and when a given engineer mission will be completed are:

- Elements specifically representing the engineer resources as to type, location, ownership, and current activity;

- Elements specifically representing the engineer mission requirements as to type, location, responsibilities, and timing;

- A mechanism for determining engineer resource capabilities, especially linking resources with task performance to specify how many resources and how much time are required;

- A method for determining responsibility and capability for mission performance;

- A method for rank-ordering multiple mission assignments so the most important missions are done first;

- A method for allocating resources to a mission and determining how long those resources will be unavailable for other missions.

Terrain Features and Facilities Altered by Engineers

The engineer task performance activities require objects to receive them, so the list of essential elements must include the representation of the terrain features and facilities that are altered during the performance of engineer tasks. These are:

- Man-made obstacles like minefields and anti-tank ditches,
- Natural/cultural obstacles like rivers and embankments,
- Prepared positions,
- Bridges,
- Roads,
- Combat trails,
- Air facilities like runways and hangars,
- Combat Service Support facilities like warehouses, supply depots, and ports.

These terrain features and facilities serve as the link between the engineer effort and the engineer effect.

Lesson Learned: The representations of the terrain features and facilities that are the focus of engineer work are the keys to how well the model can capture both the capabilities and the contributions of engineers.

This lesson became very apparent during the efforts to enhance VIC. Two examples will illustrate the importance of creating specific entities for each of the terrain features and facilities altered by engineers.

VIC's original engineer module represented the engineer task of breaching a line obstacle but had no entity to represent the breach itself. Line obstacle breaches (bridges on rivers as well as breaches of tank ditches) were represented by changing the sign (from "+" to "-") of an attribute of the obstacle section containing the breach. Such a mechanism for representing a bridge offers no way to distinguish bridges of different types or capacities and no way to place multiple bridges on a single section. A breached obstacle section would be considered breached along its entire length, yielding bridges as wide as the river section is long; no matter where a unit might encounter the section and no matter how long the section, a bridge would be available for immediate use. The engineer representation needed a way to distinguish between different types of bridges, both for the effort required to build them and the effect of each specific type on unit movement, and a way to put multiple bridges on a single section. The solution was to create a breach point entity with location and capacity attributes and to allow each line obstacle section to own a set of breach points.

VIC's original engineer module also represented the preparation of positions to protect against direct and indirect fire. The resulting position was represented as a level of preparation attribute for all unit path points within a certain distance of a location specified in scenario data. The engineer representation needed a way to distinguish the different levels of engineer effort required to prepare positions for units of different types and sizes. The solution was to create position prototypes associated with unit echelon and type and to create specific prepared position entities to be placed on the terrain independent of maneuver unit paths.

Figure 5 illustrates the ideal relationships between the three elements of the combat engineer representation; a balanced representation of engineer effort and engineer effect with the associated terrain feature serving as the pivot point between the two.

A Basic Structure for an Engineer Representation

This section presents the basic objects, relationships, and processes of a combat engineer representation that offer the flexibility and analytic control discussed earlier. This is the structure used for the engineer module of Vector-In-Commander.*

The Key Abstractions

Though the discussion thus far has alluded to the need for a high level of abstraction and simplification in the model design process, the specific top-level decisions that allow the engineer structure to be a flexible, user-controlled tool have not been identified. They are:

1. Use the same engineer task performance process for all task types.

* The reader may wish to refer to VIC's documentation and to technical reports by USACERL, the Waterways Experiment Station, and the Engineer Studies Center regarding the Engineer Model Improvement Program and the enhancements to VIC.

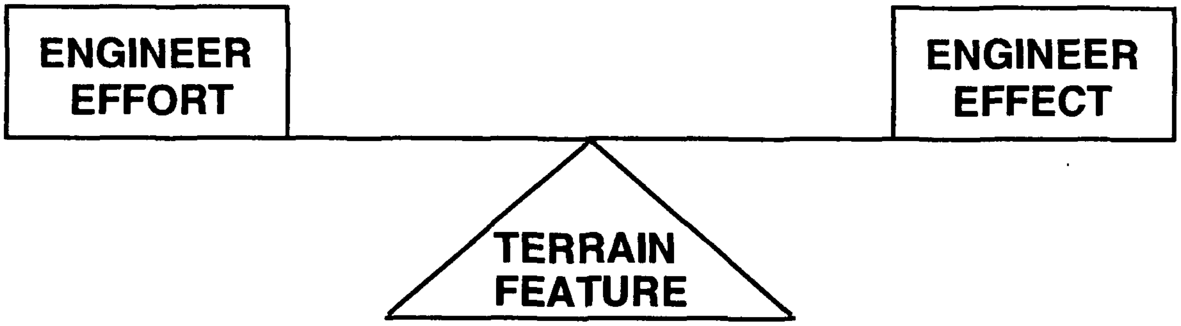


Figure 5. The Ideal Relationships for a Combat Engineer Representation.

This uniformity of structures and processes allows placing missions of all types on the same list to compete for the same resources. It also facilitates the use of the engineer technique structure discussed below and the formation of task-organized work units if the decision is to simulate the complete task performance process. In addition, this generic approach simplifies the user's requirements for understanding the module's functions and for configuring input data to take advantage of the flexibility.

2. Use the engineer technique structure to link engineer resources with specific tasks performed on specific feature types and specify all three—resources, techniques, features—in the scenario data.

Input data is structured to allow the model user to describe the terrain feature to be altered by engineers, the resources available, and the requirements in resources and time to perform the task. In other words, the code of the model provides a procedural shell that adapts to the level of detail chosen in the scenario data. This allows the engineer resources to be modeled as individual items, or as task organized teams, or as available manhours/bladehours/etc., or even as a mixture of all three. Similar flexibility exists for the terrain features altered by engineers and for the techniques they use to complete each task.

3. Link capability to perform a task with ownership of resources required for the use of a suitable technique.

This decision links ability to perform engineer task at the resource level rather than at the unit level, which is the way VIC's earlier engineer representation was structured. This automatically takes care of the problem of what to do with maneuver unit capabilities to perform engineer tasks because capability is linked to what a unit owns rather than what type of unit it is. This also opens the way for allowing the task performance process to be simulated or calculated, leaving the choice to the scenario developer. This idea is explained below under the explicit/implicit representation of task performance.

4. Specify engineer units only at the headquarters level in the unit data base with internal processes to task organize work units.

This allows a very natural structure to the input data in describing engineer units. The scenario developer may actually use a standard Table of Organization and Equipment to build the data set, and excursions on a scenario are easily altered to change force structure or resource availability.

5. Link direct/general engineer support wiu an engineer unit's superior.

This gives the scenario developer the ability to organize engineer units in a way that models both direct support and dynamic task organizing of a work unit using equipment from several units to accomplish a single mission. This concept is explained below under the section on engineer units.

The Engineer Objects and Relationships

Figure 6 illustrates the basic objects of the engineer module and the relationships and processes that link them to each other. This section discusses the objects and their key attributes; the next section presents the flow of each process.

Engineer Tasks. The individual engineer tasks to be represented in a combat simulation are the internal anchor for the entire structure. USACERL researchers have not found a way to avoid coding the model's engineer task list and then using a pointer or reference number for each task to link what must be done in a mission with a way to do it and with an appropriate effect when the mission is complete. The fixed task reference number allows all of the other entities that must track a specific task to do so with ease. Each engineer mission, job, and technique has an associated task attribute, and the procedures for generating a mission and registering the effect of task completion select appropriate actions according to that task reference number. In VIC's case, the SIMSCRIPT II.5 define-to-mean feature allowed the implementation of a simple numbering system for the tasks, easing both the gradual addition of new tasks as model capabilities grew and the reference to each task in the code by its descriptive name.

TASK	REFERENCE NUMBER
Emplace obstacle	1
Breach obstacle	2
Improve breach	3
Destroy breach	4
Prepare position	5
Build combat trail	6
Maintain road	7
Repair road crater	8
Repair runway	9
etc.	

A list such as the one above must be supplied to the scenario developer to ease references in the data to techniques for each task type.

Terrain Features. Having decided which engineer tasks are to be represented, the next step is to define the types of terrain features to be altered by each task and then determine both what engineers need to k ow about each feature to assess the level of effort required for the task and what units encountering the feature need to know to assess its effect. A brief outline is given below with no attempt to be exhaustive in listing either terrain features or attributes. What is perhaps most important is to have a data structure that allows a general description by type and then designated instances of actual features of each type. For flexibility, the scenario data should determine the number of different types being represented in each terrain feature category and the characteristics that distinguish both the effort and effect associated with each type.

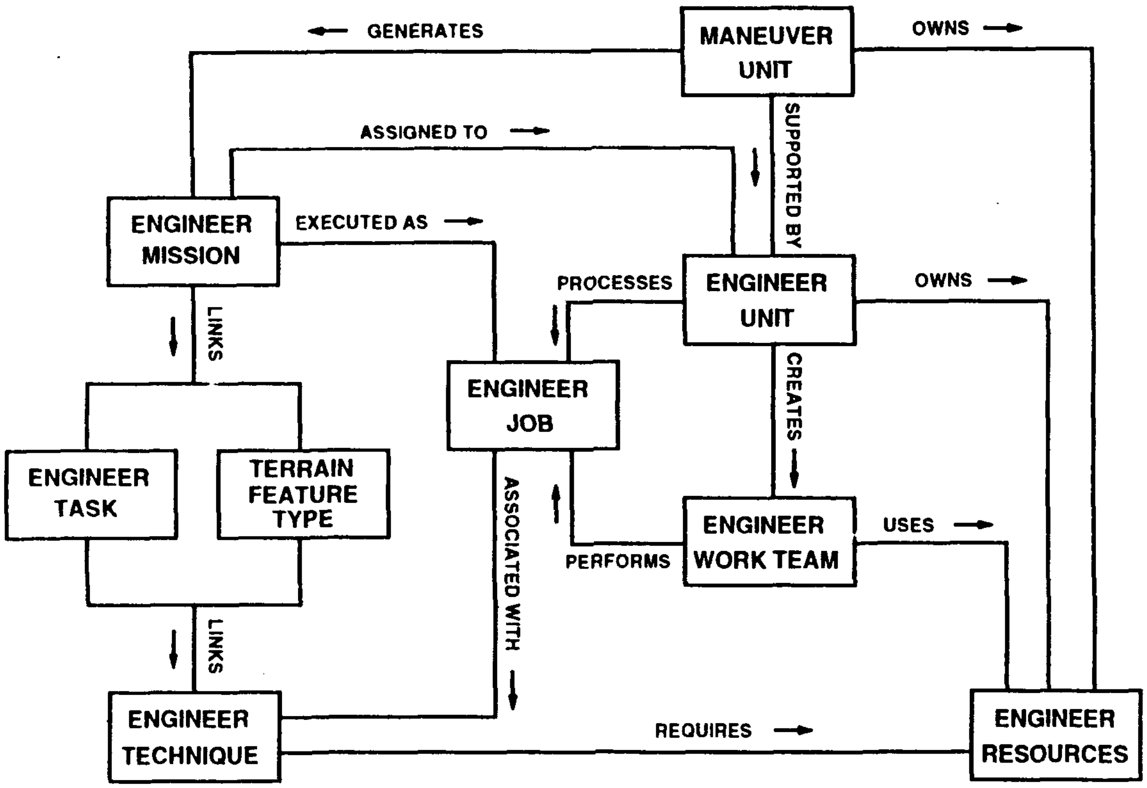


Figure 6. The Essential Objects, Relationships, and Processes for Representing Engineer Effort.

The set of terrain feature objects is very model-dependent, being at the core of the basic decisions a model designer makes about how units will interact with each other and with the surrounding terrain. Those basic design decisions have a great effect on the range of questions to which the model may be applied. The lesson bears repeating that the representations of the terrain features and facilities that are the focus of engineer work are the keys to how well the model can capture both the capabilities and the contributions of engineers.

Obstacles. Engineers are required to emplace and/or breach a large number of different kinds of obstacles, both man-made and natural. A combat model may have an obstacle representation that is specific to each type of obstacle, or it may take a more general approach in which all obstacles are seen as sharing certain basic characteristics that determine both engineer effort and effect. Thus the model designer may choose to create unique obstacle structures for each type of obstacle and unique engineer tasks for emplacing or breaching them. The task list might expand from "emplace obstacle" to include a number of more specific tasks: "emplace minefield," "emplace tank ditch," "emplace area obstacle," etc. Another approach might abstract the idea of an obstacle as an object on the terrain which covers a specified area that affects unit speed and survivability as a unit moves through it, and is linked with other obstacles to limit an encountering unit's reaction alternatives, especially bypassing or breaching. Regardless of how the obstacles are represented, the design must strive to capture the commander's intent for them. This forces the obstacle representation to include more than just the independent effect of the obstacles themselves since obstacles are always a part of a larger plan.

Prepared Positions. The resolution of the model will determine whether positions are represented at the weapon or unit level, and the amount of preparation effort is relatively easy to link to the particular position type. Perhaps the most difficult parts of portraying position preparation are the division of effort between engineer and nonengineer elements and the accurate measure of continuous improvement.

Preparation of positions is the only engineer task that consistently proceeds on an "as-time-permits" basis, requiring a continual updating of its completion instead of allowing a one-time registration of effect. This task is also the only one that is primarily the responsibility of the maneuver force, requiring a task-specific accounting of labor distribution that is not usually required by other task types.

Roads. Road travel is very different from cross-country movement, and highly-mechanized armies are dependent on having some type of road infrastructure in all but the forward areas. So a combat model must necessarily portray travel by road for all units and will most likely represent the two types of ground movement with separate algorithms. Road travel affects a unit's formation which in turn affects the unit's vulnerability and firepower; and road travel affects a unit's speed, especially at night. The most important factors about a road derive from how fast a unit can travel on it, which is dependent on surface composition, width, damage level, and congestion. Of most interest to engineers is the damage level, both from intentional obstacles like craters and demolished bridges and from normal degradation as a result of heavy equipment usage.

Bridges. On the battlefield, bridges come in two varieties: bridges that are a part of the road infrastructure of the theater of operations and temporary assault bridges emplaced by units moving cross-country. One might be inclined to think that a bridge is a bridge and that both types should have the same underlying structure. But these two types of bridges are different in almost every way. Bridges on roads are of no concern to road travelers so long as the bridges are intact; speed and unit formation are unaltered. Assault bridges are another matter entirely. Units moving cross-country must synchronize movement with engineer bridge emplacement. Such units are vulnerable to attack on either bank or in crossing, and both unit speed and formation are considerably altered. In addition, assault bridges such as armored vehicle launched bridges tend to be of the retrievable variety, requiring synchronization with follow-on forces to produce a smooth forward movement of the train. The two bridge types are conceptually similar in that units on the move should be able to recognize either type as a means of crossing a river. However, that is where the similarity stops. Each of the two bridge types requires its own implementation both for engineer effort to emplace and remove and for the effect of its presence or absence on unit movement.

Engineer Missions. The process of generating missions for engineers and moving into the engineer module to assess task performance capabilities is the major interface between the nonengineer and the engineer specific parts of a combat model. Missions are generated both from the scenario data and from the evolving situation as the simulation progresses. To cross the boundary between nonengineer and engineer sections, a mission entity is required to track the crucial information: the task to be done, the type of feature to be altered, a list of the actual features that are to be altered, the location of the work site, the maneuver unit ordering the work, the time the work was requested, the earliest time the work may begin, the latest time the work may be completed, and perhaps an indicator of how a technique is to be chosen for doing the work. Each opposing side has its own list of missions to process. By using reference numbers or pointers for the task type, feature type, and actual features, all types of missions can be tracked with the same data structure, and the mission assignment process can move from a task-specific mission into a generic processing of the work which need not look again at the specific type of task being processed until the work is complete.

The side's mission list represents the unassigned engineer work to be done. Periodically, each mission on the list is examined to see if conditions are right to assign it to a specific engineer unit for performance. The conditions that are checked in assigning a mission are the intuitively natural ones concerning timing, engineer support capabilities and locations, and the availability of a suitable technique. If the mission is assigned, it is removed from the mission list to become an engineer job, which is processed by simulating the task performance. If the mission is not assigned, it remains on the mission list with the possibility that the requesting unit itself will perform the work. Using a nonengineer unit's engineer resources to calculate the mission completion time yields an "implicit engineer representation" because the engineer capability is assessed without an explicit engineer presence and without the simulation of task performance.

Engineer Assets. The engineer assets provide the structure for representing the equipment resources required to perform engineer tasks. Among the objects of the simulation, the engineer assets are usually classified as weapons, primarily because a unit's effectiveness, strength, viability, logistics requirements, and combat focus all revolve around the weapons it owns and the weapons it must oppose. In an item-level simulation, this representation is consistent and efficient, using one data structure to portray not only engineer capability but also target value and vulnerability. But in aggregated simulations, problems arise with this dual interpretation. Weapons are not represented as individual items but may be represented as groups of weapons of a certain number and type or in even more aggregated terms such as total mass equivalents. This representation leads to various levels of resolution mismatch between the engineer resources and their dual role as weapons, with the severity of the mismatch depending on the level of weapons aggregation and how much engineer detail the model is required to track.

The resolution mismatch occurs because the engineer task performance is quite dependent on the functioning of a small number of individual pieces of equipment whereas aggregated attrition produces fractional levels of damage that may not be easily attributed to individual items. A technique for preparing a defensive position, for example, may require the use of 2 dozers, while an engineer unit may have 3.76 surviving dozers of its original 4. How many position preparation missions can be allocated at once? Has all of the damage been sustained by one dozer or is the damage spread evenly across all 4 original dozers? Even though the attrition calculation may be based on an even distribution of damage, attributing all of the damage to a single item may be a more realistic interpretation. And with that interpretation, what is the work capability of .76 dozers?

In a low resolution approach, both engineer equipment holdings and technique equipment requirements can be specified with single floating-point attributes. A unit may have X dozers and a technique may require Y dozers. If X >= Y, then Y dozers are allocated and X - Y dozers remain. If the task performance process is simulated and the Y dozers sustain Z losses, then the unit's dozers will number (X - Y) + (Y - Z), or X - Z, when the equipment returns to its base.

The above "floating point" approach is not sufficient, however, if the model is to provide any of the following additional features for assessing engineer effort:

- Task organizing a work unit with equipment pooled from several units,

- Intermittent required rest period so the engineer equipment/manpower does not work continuously,

- Item-level attrition of engineer equipment,

- Tracking of the activity of each individual equipment item.

All of these features ultimately force the model designer to confront the question of how best to represent engineer resources. For example, do 10 dozers at 90 percent effectiveness yield the same engineer capability as 9 dozers at 100 percent effectiveness? In the floating-point approach, both would be recorded as 9.0 dozers, but the 10 dozers can be physically configured in ways that 9 dozers cannot be configured. Given engineer dependence on small numbers of equipment items, the difference is significant, especially for critical items.

In the VIC engineer module, three separate entities track engineer equipment. A weapon group tracks each type by a floating-point amount. An integer array tracks each type by integer count of individual items. And a unit inventory tracks each equipment item, linking the integer count to the weapon count by an equipment effectiveness level (1.0 indicating no damage, and 0.0 indicating destroyed). For example, a unit's dozer weapon group may indicate 5.6 dozers, its integer array may indicate 7 dozers, and its inventory would contain 7 dozers with activity "available." The reconciliation of these numbers comes with the 7 inventory records for the available dozers. These records may indicate that the first 3 dozers have an effectiveness level of 0.6 each, that the next 3 dozers are at effectiveness level 1.0, and that the seventh dozer is at effectiveness level 0.8. Thus, 5.6 = (3 * 0.6) + (3 * 1.0) + 0.8, yielding 7 dozers at various levels of damage.

This more detailed accounting of the engineer equipment is not without difficulties. Simulating the engineer task performance process allows equipment to move about the battlefield apart from the engineer unit that owns it. Damage to a work unit must be allocated to the individual equipment items, and all recordkeeping devices must be made to agree. A damage threshold must be established for each type of equipment so that decisions can be made regarding when an individual equipment item has reached a point where it can no longer function, and a method must be devised to degrade the performance of damaged equipment whose level of effectiveness is above the threshold. Aside from the fact that all of this must be done with unstable floating-point arithmetic, many problems must be resolved when relatively continuous functions control step functions, which in turn control decision processes.

In spite of the difficulties associated with this triple bookkeeping, the integer equipment array and unit inventory simplify other aspects of the engineer process. Each engineer technique has a required equipment array similar to the unit array, so a large portion of the decision about a unit's capability to use a technique rests on simply comparing arrays.

Techniques. For each side and task type/feature type, input data specifies a set of techniques by which the corresponding work may be done. In essence, the technique data identifies the resource and time requirements for accomplishing a mission of a certain task and feature type. The general nature of the technique structure provides a tremendous amount of flexibility. By avoiding any linkage with engineer units per se, the technique data can be used by engineers and nonengineers alike. This opens the way for two very important features of the engineer representation: accounting for nonengineer capabilities to perform engineer tasks and allowing a calculated engineer effort without an explicit representation of engineers or their processes. Both of these features are accomplished with one set of algorithms called the implicit engineer representation.

Accounting for nonengineer capabilities is an important part of properly assessing the engineer contribution. Indeed, many engineer tasks may be performed by nonengineer units using equipment that would not be strictly classified as engineer equipment. For example, armor units use plows and rollers to breach minefields. The technique data, combined with the listing of tank plows and rollers as engineer assets, may include a minefield breaching technique that may only be used by armor units because no engineer units own plows and rollers. Scenario data may list plows and rollers as a part of a maneuver unit's weapon inventory. When such a unit encounters an enemy minefield, it automatically generates an engineer mission to breach it. Failing in the attempt to assign the mission to an engineer unit, the maneuver unit itself may then consider its own capabilities by comparing a list of its uncommitted engineer assets with the list of assets required for a suitable technique. If a match is made, the technique data holds the information necessary to calculate how long the mission will take and what unit resources must be committed or expended in the process. The calculated time is used to schedule the completion of the breach, and the completion event keeps track of the assets committed to the task.

This method of accounting for nonengineer capabilities may be expanded in interpretation without changing any of the process to yield a new way to calculate engineer capability. Rather than explicitly creating engineer units to support nonengineers, the scenario developer may choose to place the engineer assets in the weapon inventories of the larger maneuver units. The explicit mission assignment process will rapidly determine that no engineer unit is available and will assess the maneuver unit's capability. On finding such a capability, the mission completion time is computed from the technique data using the maneuver unit's uncommitted assets and on-hand supplies and the conditions at the work site. As indicated conceptually in Figure 7, the implicit engineer representation bypasses the detailed engineer simulation of task performance but still takes the crucial factors into account in determining if and when a particular mission will be accomplished. (Compare Figures 6 and 7 for explicit engineer representation.) An added benefit is that the explicit and implicit representations are compatible with each other and with the representation of nonengineer capabilities as well. Yet each representation has a distinctly different approach to assessing the capability of accomplishing an engineer mission, with the choice of which to use left to the scenario developer. Internally, the process moves automatically from consideration for explicit performance to implicit calculation.

The order in which the techniques are listed in the input data determines the order in which they are considered for use, but other criteria may determine how a technique is chosen. Three possibilities exist in VIC: choosing the first technique in the list that fits the conditions, choosing the fastest suitable technique, or choosing the suitable technique with the most effective end result. The manner in which a mission was generated determines which of the three criteria is used.

Each technique has attributes that help to determine its suitability for use in certain situations. One attribute indicates the echelon of units that normally use the technique, so that an attempt can be made to match units with techniques appropriate to their level. Another attribute indicates whether the technique is applicable to work in the rear or to work at the forward edge of the battle area (FEBA). This latter attribute could be a simple flag or a distance threshold.

Each technique has a point of organization, which is used when the engineer process is being simulated. Since preemption of committed equipment and pooling of equipment from other engineer units are both allowed, this attribute gives the scenario developer the option of choosing one of two work unit organizational methods. For a technique with a headquarters organization point, all of the equipment moves first to the responsible headquarter's location and forms one work unit to proceed to the work site. For a technique with the work site as the organization point, each set of collocated equipment being used for the work moves directly to the work site and merges into one unit there. Work begins as soon as the equipment required for the current segment is at the site. In addition, when the organization point is at the work site, equipment leaves for its next destination as soon as it is no longer needed at its present site. When the organization point is the headquarters unit, the equipment assigned to a particular phase of the work stays together until the phase is complete.

The key structure for tracking the task performance through a multitude of steps is the technique segment. The segments mark discrete points at which an effect can be measured and a segment duration time can be predicted. The segments may be part of a continuous work effort that ties up all needed equipment for the duration or a disjoint set of activities using different equipment in phases that only tie up the equipment used in that phase. Each segment has an attribute that indicates the action to be taken when that segment is finished. The end actions may be (1) continue on to the next segment (continue), (2) end the work of the current work unit and assign a new work unit to do the next segment (break point), and (3) register the completion of the work (end of job). A phase is defined to be from the current segment to the end of the first succeeding segment with an end action that is a break point or end of job. Dividing a job into segments allows a job to be preempted by a higher priority job but still have partial effect and allows work units to work in the assault, where fluctuations in available equipment alter the duration or cancel a job. Also, equipment used only at the beginning of the technique may be released for other jobs before the present job is finished, and interrupted jobs can have a measured effect and can be resumed without starting from the beginning.

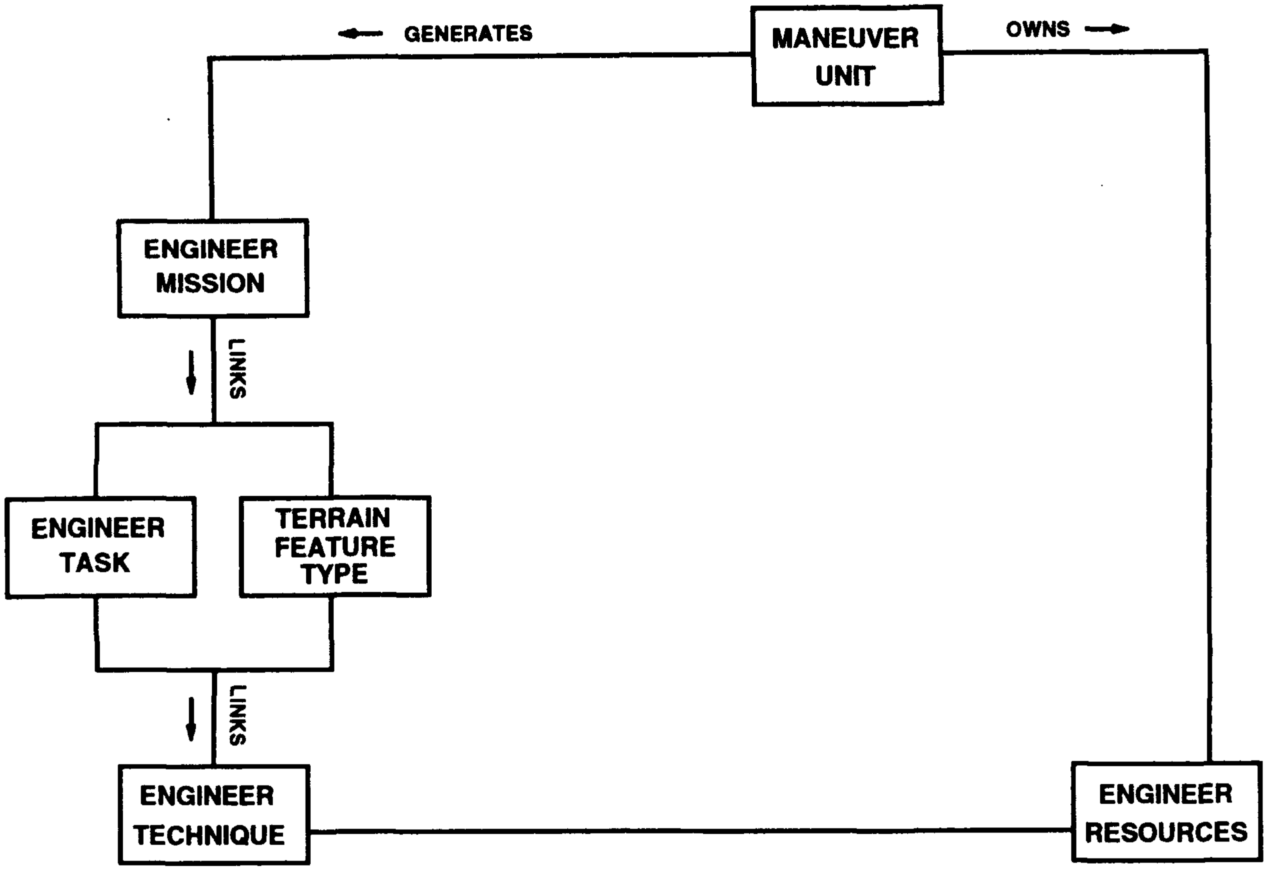


Figure 7. Conceptual Diagram of Implicit Engineer Representation.

Each segment has an attribute for its end effect, which indicates the fraction of the work complete at that stage, and an attribute for its intrasegment effect to reference a function for computing how much of the work is complete if the segment is interrupted before it is finished. USACERL used four functions in VIC: step, linear, skew-right, and skew-left. For a task such as building a bridge, for example, a step function might be used to indicate that the effort had no effect unless the segment was finished. For a task such as emplacing a minefield, a linear function is appropriate if working x percent of the time produces x/100 of the minefield. If a minelayer is used, a skew-right function is more appropriate to capture the idea that the work starts slowly but finishes quickly. With x percent of the working time complete, $(x/100)^2$ is the fraction completed. For a task such as breaching an obstacle, a skew-left captures the idea that more than half of the effect would be completed in the first half of the work time. With x percent of the working time complete, $(x/100)^{1/2}$ is the fraction completed.

Each technique has a list of required equipment that determines the type of equipment to be used and the preferred and minimum number of pieces of each type. This list is used first when it is compared with a unit's assets to determine whether or not the unit can perform a task by this technique, with the criterion being that the unit own the minimum required number of each type of equipment. If this technique is chosen, the required equipment list is used to determine when an asset is needed at the work site and to determine job segment duration if the preferred amount of equipment is not available or assets have been damaged. The segment duration is based on the availability of the preferred amount of equipment. If less than the preferred amount of equipment is available, the duration is increased proportionally. If equipment levels fall below the minimum required, the mission is discontinued.

When equipment is damaged or preempted, the time to complete the current and future segments is adjusted to reflect the loss of the equipment. If only one type of equipment is lost, completion time for a segment is adjusted to reflect the capacity of the current number based on the duration and capacity of the original number. If equipment of different types is lost, the previous computation is done for each type of equipment and the maximum duration time is used.

A base preparation time for each type of required equipment is used to determine prejob activity time for the work unit. If any of the equipment requires base preparation time, the maximum of those amounts is used to determine the departure time.

Each technique also has a list of required supplies for the work. The mission assignment process verifies that an engineer unit is authorized to receive such supplies before the unit can be assigned to do the mission. If the work is assigned to an engineer unit, that unit must have the supplies on hand before it can form a work unit; the supplies are transferred to the work unit to be expended at the site at the appropriate time. If the work is done implicitly, the maneuver unit performing the work must have the supplies on hand; and those supplies are expended as soon as the work begins.

Units. Units in a combat simulation generally have the following attributes:

- side,
- name,
- type,
- echelon,
- superior,
- location,
- destination,
- a path to get to destination,
- designated area of interest for each path location or a radius of interest,
- specified radius of control or firing range,
- a mission,
- equipment/personnel inventory to keep track of number, type, and damage level of assets, and
- supply inventory to keep track of number, type, amount on-hand, and reorder threshold of supplies.

The most logical approach to representing engineer units is to begin with this general description and to use as much of it as possible in a natural way to build the engineer processes: the superior determines the support relationship, the area of interest defines the area of possible work sites, the unit's location and path affect travel times to work sites, and unit inventories for equipment and supplies determine available resources.

The attributes of the **engineer headquarters unit** are the key elements in the explicit representation of engineer task performance. The headquarters unit holds the engineer resources and is responsible for all explicit engineer work. It actually appears in the model's unit database as one of the units in the organizational structure of a side, and its attributes are set by the scenario developer. Input data controls which unit it supports (its superior), what resources it owns (its inventories), where it is at any given time (its location and path), and where it will send crews to work (its areas of interest). All of these attributes affect the level of resources available for a given task and the amount of time required to complete it.

The simulation of the task-performance process is accomplished by creating a second type of engineer unit, the **engineer work unit**. The work unit is a task-organized unit created internally to perform a specific set of jobs using equipment and supplies transferred to it by the headquarters unit responsible for the jobs. The work unit simulates the process of moving equipment and supplies across the battlefield from base to work site, performing the work, and moving back to the base again.

To address the different types of command and support relationships, the representation requires a third type of engineer unit, the **engineer staff unit**. A staff unit is a particular type of headquarters unit, one that has a nonengineer superior. Staff units are identified internally from the organizational structure of the unit database and are created as entities in their own right, separate from the headquarters unit entity. In the organizational structure given in Figure 8, engineer headquarters units ENG-1, ENG-A1, and ENG-A2 are staff units. ENG-A3 and ENG-A4 are not staff units. With this organizational structure, all of the jobs assigned to ENG-A1, ENG-A3, and ENG-A4 are prioritized on one list by ENG-A1, and the equipment of all three units may be pooled to work on a single job. This structure allows ENG-A1, for example, to hold equipment in reserve but to have it automatically available if its subordinates are over-committed. In addition, ENG-A3 can be assigned a general support role with MVR-A3 by specifying the same area of interest for both units.

The staff unit entity plays an important role in many of the engineer processes. The mission assignment process chooses a responsible staff unit first, assessing the capability and support assignments of each staff's entire command chain and choosing a responsible staff before moving down its command chain to assign the task to the unit in the chain closest to the work site and at the appropriate echelon. Staff units always directly support their immediate superior, and they prioritize and process the work of their command chain according to their superior's activities.

Jobs. Once an engineer mission is assigned to a particular engineer headquarters unit, a new entity called the engineer job is created, and the mission entity is destroyed. The job structure has many more attributes than a mission since it must keep track of work progress, various timing elements, and work unit assignments. The job structure has attributes for:

- type of task
- type of feature,
- pointer to actual feature being altered,
- technique being used,
- size of job if appropriate (number of mines for emplacing and clearing of minefields, length in kilometers for emplacing linear obstacles and maintaining roads, and depth in kilometers for breaching minefields),
- current job priority,
- time mission was created,
- time first work unit was formed to work on this job,
- earliest time that work may begin on this job,
- latest time that work may be completed,
- time the current segment started,
- estimated remaining amount of time required to complete work,
- pointer to the last segment of work completed,
- time at which last segment was completed,
- pointer to the last segment for which a work unit has been assigned,
- location of the job site,
- pointer to the maneuver unit ordering the work,
- pointer to the headquarters unit responsible for the work,
- pointer to the mission for which this job is a part,
- original mission number of job if generated by input data, and
- list of active phases for this job; each active phase has,
  - first segment of the phase,
  - last segment of the phase,
  - an array indicating total number of each type of equipment required for this phase, and
  - list of pointers to work units assigned to this phase.

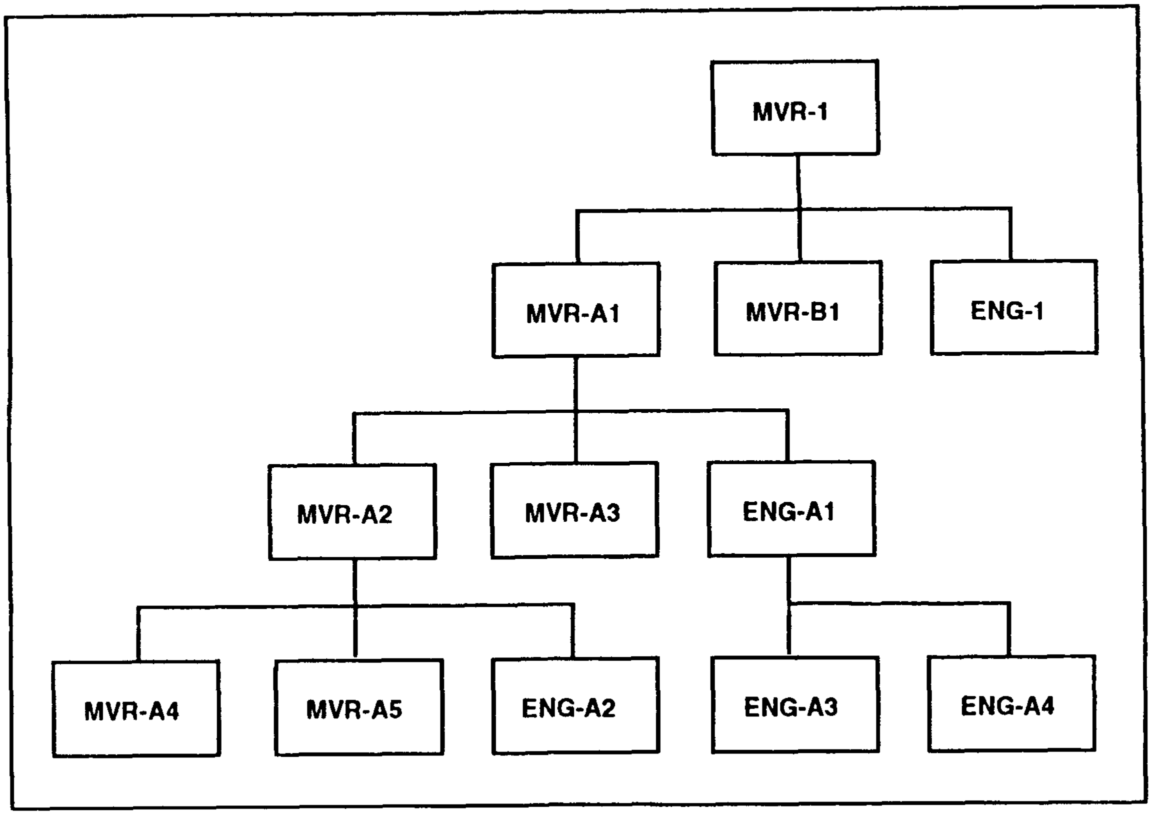


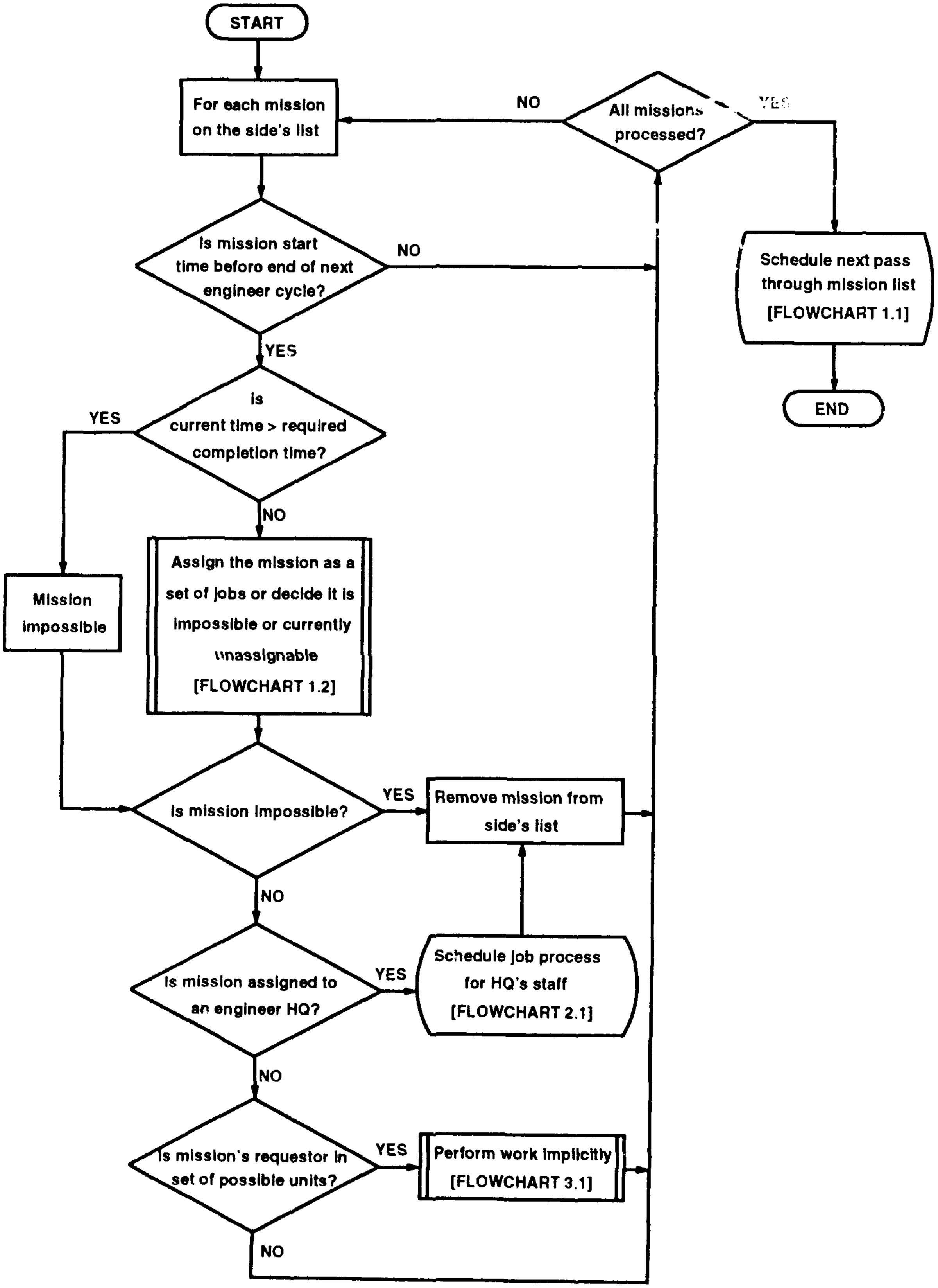
Figure 8. Template for Engineer Unit Organization.

The engineer job entity is held on either a pending or active list owned by the responsible headquarters unit. Since both the corresponding staff unit and work units also track the job, each of these units has a list of pointers to engineer jobs appropriate to their use.

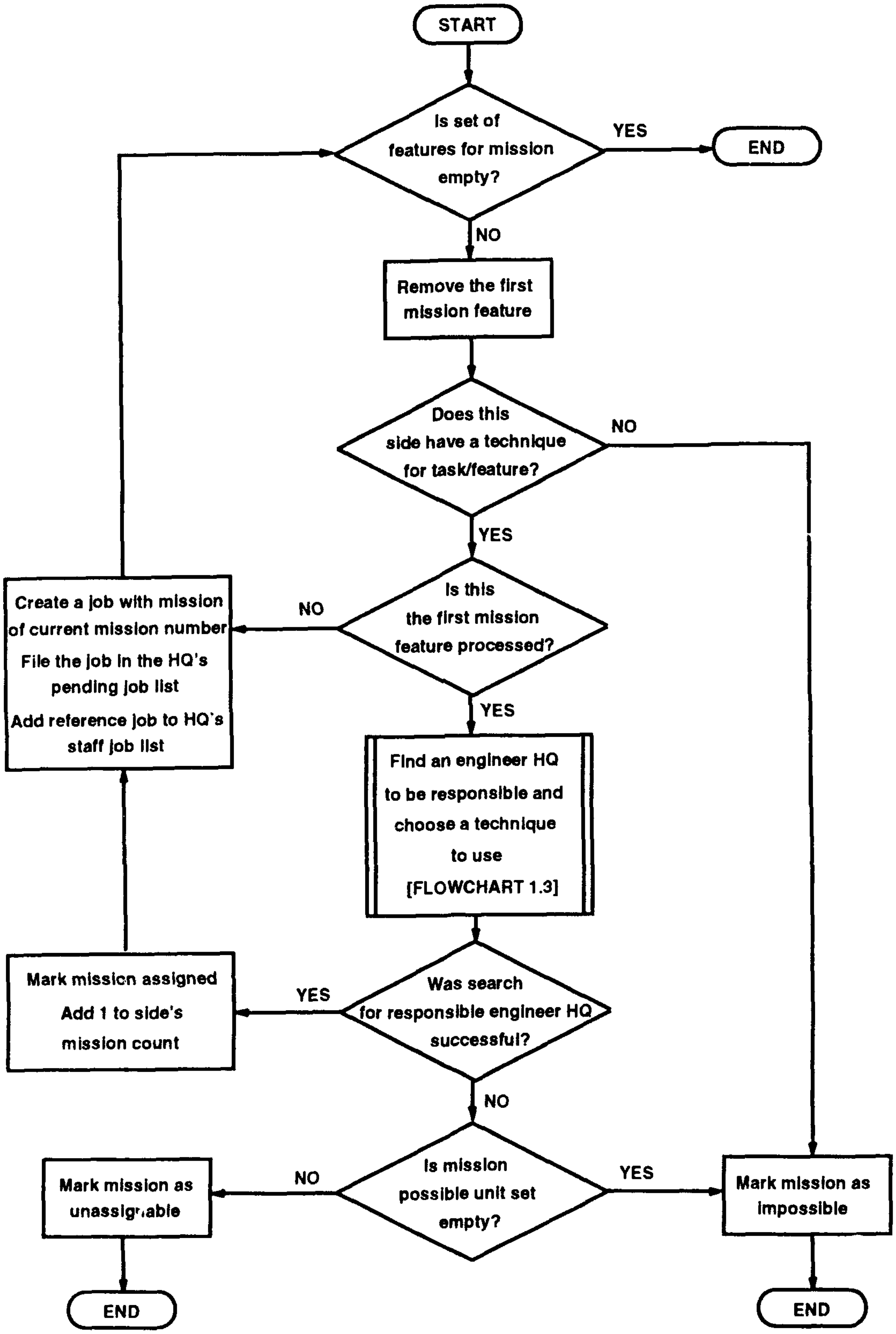
The Engineer Processes

The basic engineer processes are described on the following pages in flowchart format covering the eight basic engineer operations:

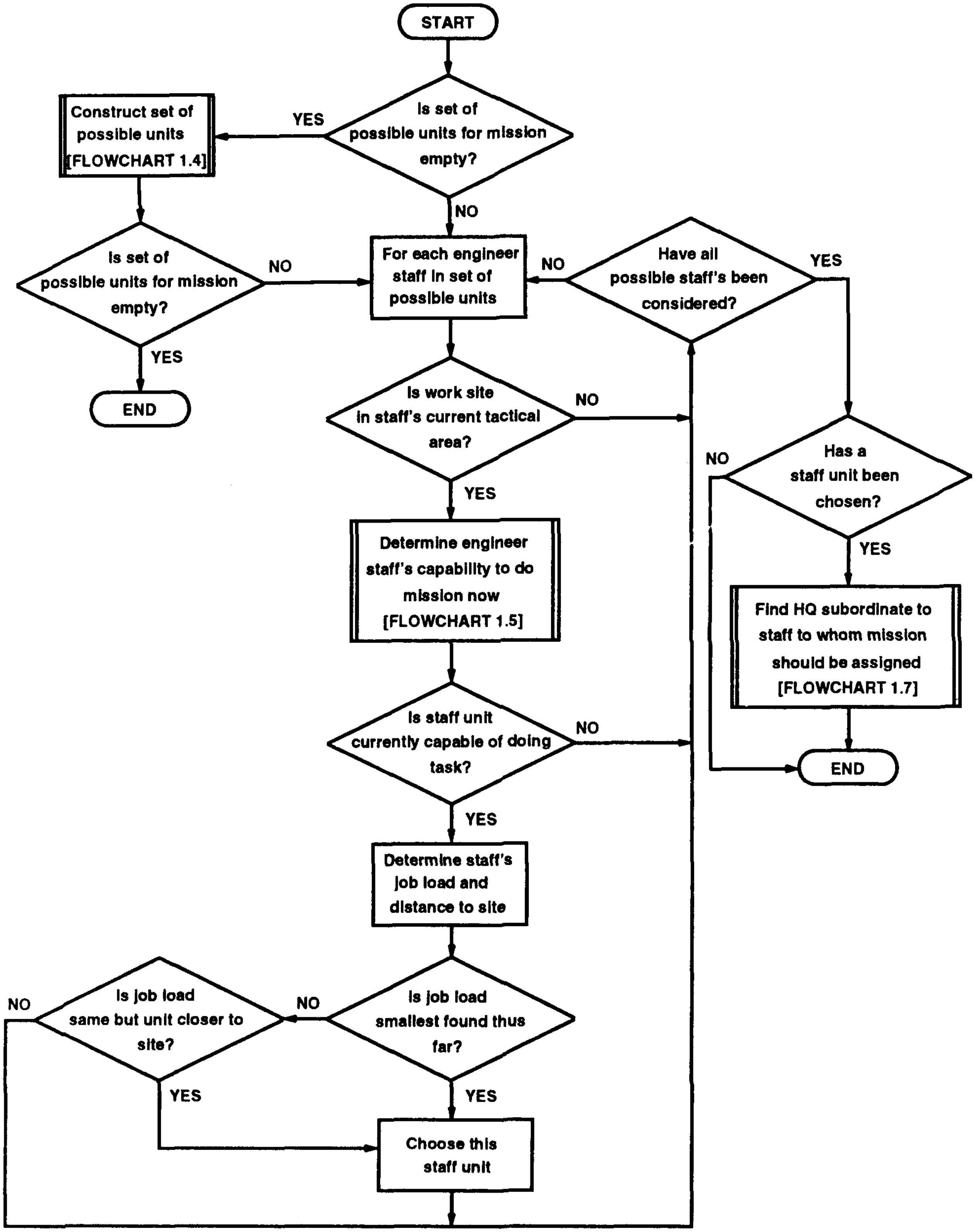
1. Assign a mission to a unit,
2. Allocate resources to the mission,
3. Perform the mission implicitly,
4. Move resources to site and begin work,
5. Complete a segment of the work and decide next course of action,
6. Discontinue work,
7. Register effect of engineer work, and
8. Register effect of implicit engineer work.


START
For each mission on the side's list
NO
All missions processed?
YES
Is mission start time before end of next engineer cycle?
NO
YES
Schedule next pass through mission list [FLOWCHART 1.1]
END
Is current time > required completion time?
YES
NO
Mission impossible
Assign the mission as a set of jobs or decide it is impossible or currently unassignable [FLOWCHART 1.2]
Is mission impossible?
YES
Remove mission from side's list
NO
Is mission assigned to an engineer HQ?
YES
Schedule job process for HQ's staff [FLOWCHART 2.1]
NO
Is mission's requestor in set of possible units?
YES
Perform work implicitly [FLOWCHART 3.1]
NO

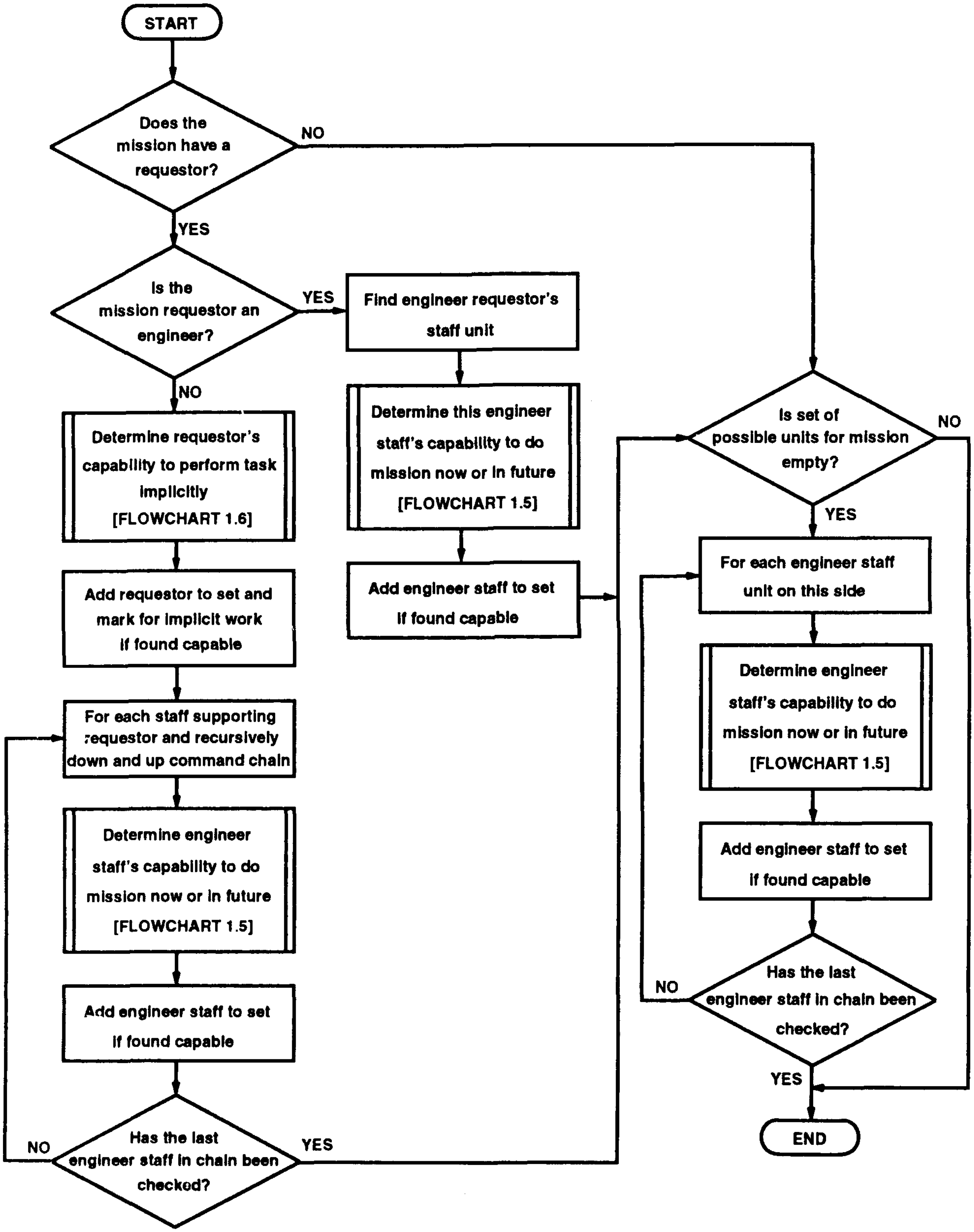
FLOWCHART 1.1 - PROCESS SIDE'S MISSION LIST


START
Is set of features for mission empty?
YES
END
NO
Remove the first mission feature
Does this side have a technique for task/feature?
NO
YES
Is this the first mission feature processed?
NO
Create a job with mission of current mission number
File the job in the HQ's pending job list
Add reference job to HQ's staff job list
YES
Find an engineer HQ to be responsible and choose a technique to use [FLOWCHART 1.3]
Was search for responsible engineer HQ successful?
YES
Mark missicn assigned
Add 1 to side's mission count
NO
Is mission possible unit set empty?
NO
Mark mission as unassig'able
YES
Mark mission as impossible
END
END

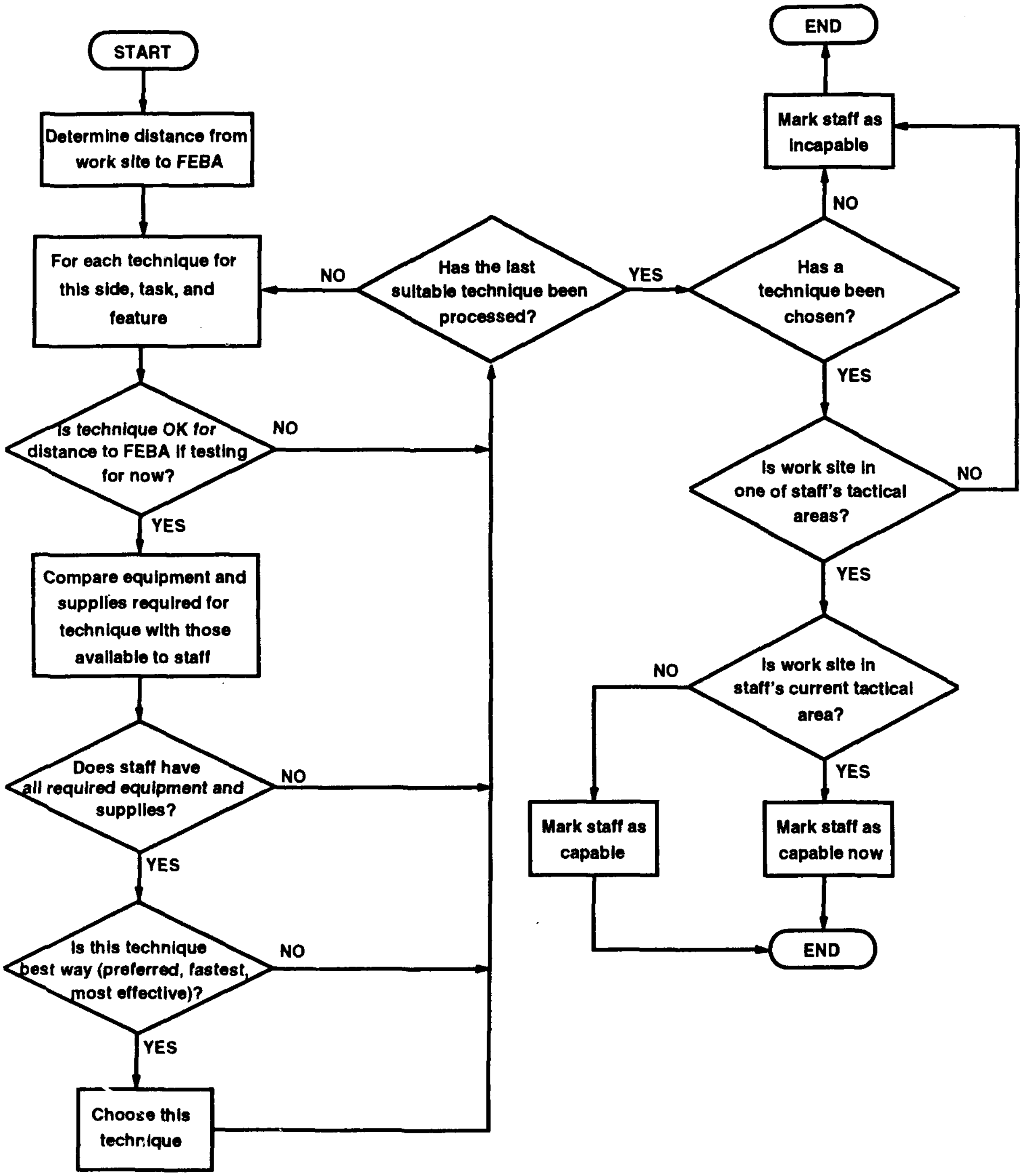
FLOWCHART 1.2 - ASSIGN MISSION OR DECIDE IMPOSSIBLE OR UNASSIGNABLE


START
Is set of possible units for mission empty?
YES
Construct set of possible units [FLOWCHART 1.4]
NO
Is set of possible units for mission empty?
NO
YES
END
For each engineer staff in set of possible units
NO
Have all possible staff's been considered?
YES
Is work site in staff's current tactical area?
NO
YES
Has a staff unit been chosen?
NO
YES
Determine engineer staff's capability to do mission now [FLOWCHART 1.5]
Find HQ subordinate to staff to whom mission should be assigned [FLOWCHART 1.7]
END
Is staff unit currently capable of doing task?
NO
YES
Determine staff's job load and distance to site
Is job load same but unit closer to site?
NO
NO
Is job load smallest found thus far?
YES
YES
Choose this staff unit

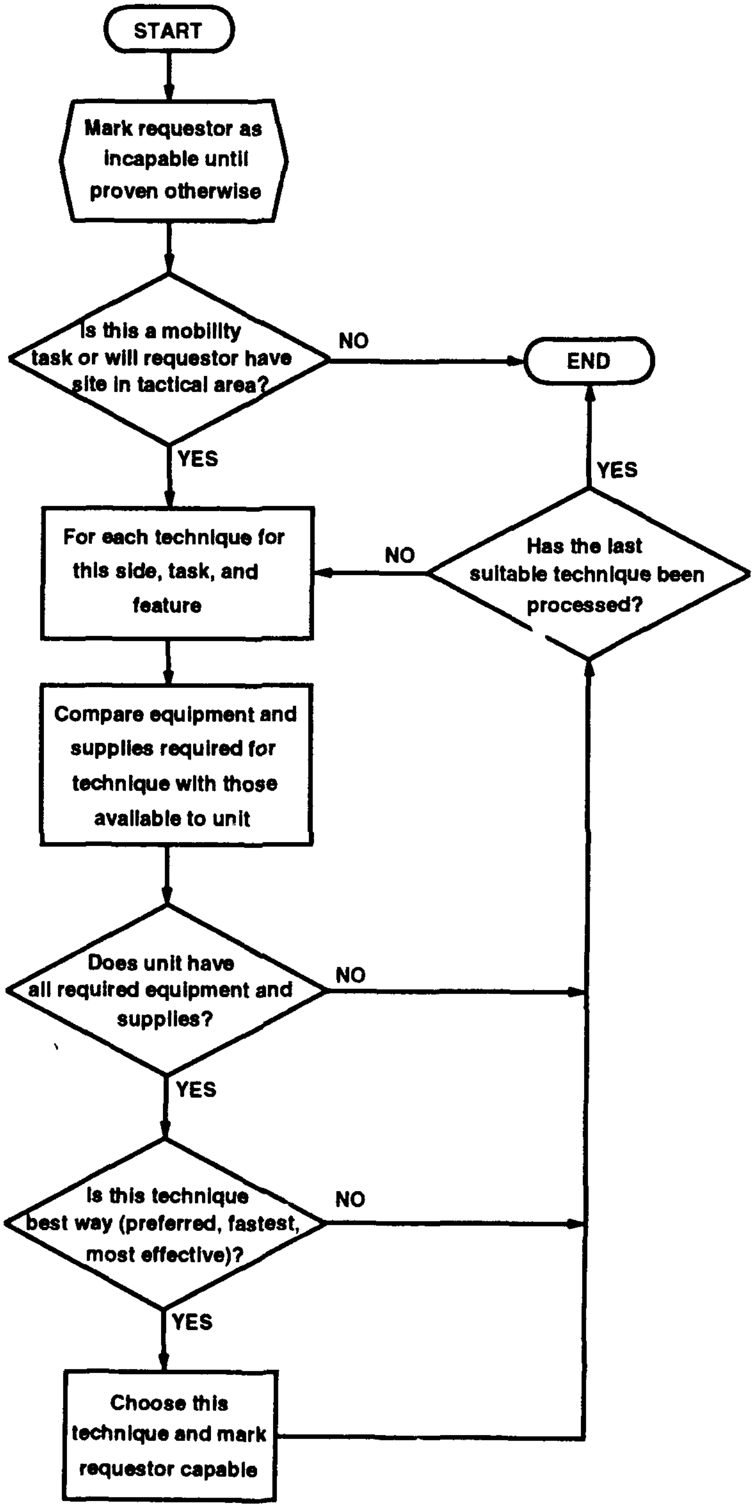
FLOWCHART 1.3 - FIND AN ENGINEER HQ TO BE RESPONSIBLE FOR MISSION


START
Does the mission have a requestor?
NO
YES
Is the mission requestor an engineer?
YES
Find engineer requestor's staff unit
NO
Determine requestor's capability to perform task implicitly [FLOWCHART 1.6]
Determine this engineer staff's capability to do mission now or in future [FLOWCHART 1.5]
Add requestor to set and mark for implicit work if found capable
Add engineer staff to set if found capable
For each staff supporting requestor and recursively down and up command chain
Determine engineer staff's capability to do mission now or in future [FLOWCHART 1.5]
Add engineer staff to set if found capable
NO
Has the last engineer staff in chain been checked?
YES
Is set of possible units for mission empty?
NO
YES
For each engineer staff unit on this side
Determine engineer staff's capability to do mission now or in future [FLOWCHART 1.5]
Add engineer staff to set if found capable
NO
Has the last engineer staff in chain been checked?
YES
END

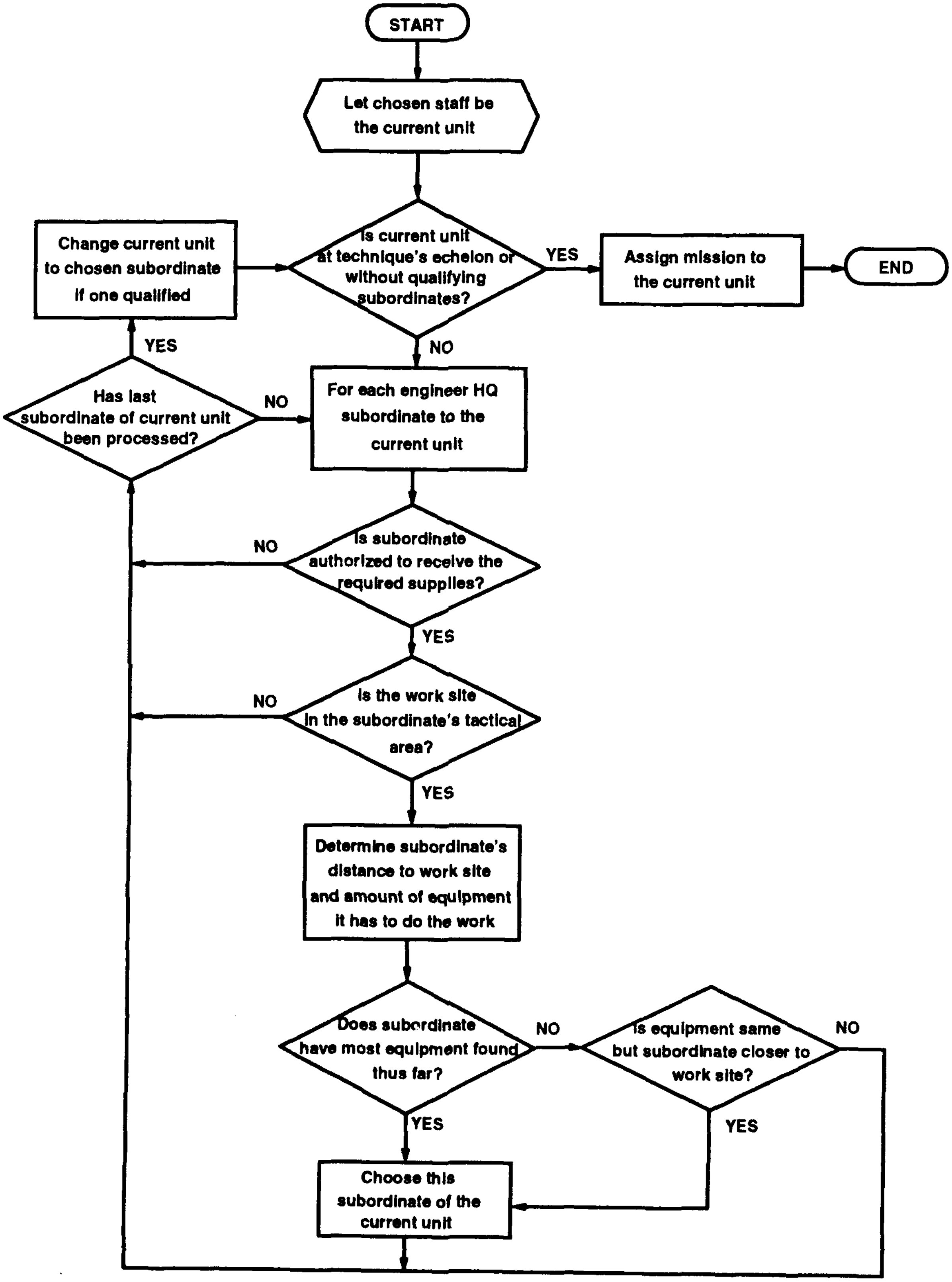
FLOWCHART 1.4 - CONSTRUCT MISSION'S SET OF POSSIBLE UNITS


START
Determine distance from work site to FEBA
For each technique for this side, task, and feature
Is technique OK for distance to FEBA if testing for now?
NO
YES
Compare equipment and supplies required for technique with those available to staff
Does staff have all required equipment and supplies?
NO
YES
Is this technique best way (preferred, fastest, most effective)?
NO
YES
Choose this technique
Has the last suitable technique been processed?
NO
YES
Has a technique been chosen?
NO
YES
Mark staff as incapable
END
Is work site in one of staff's tactical areas?
NO
YES
Is work site in staff's current tactical area?
NO
YES
Mark staff as capable
Mark staff as capable now
END

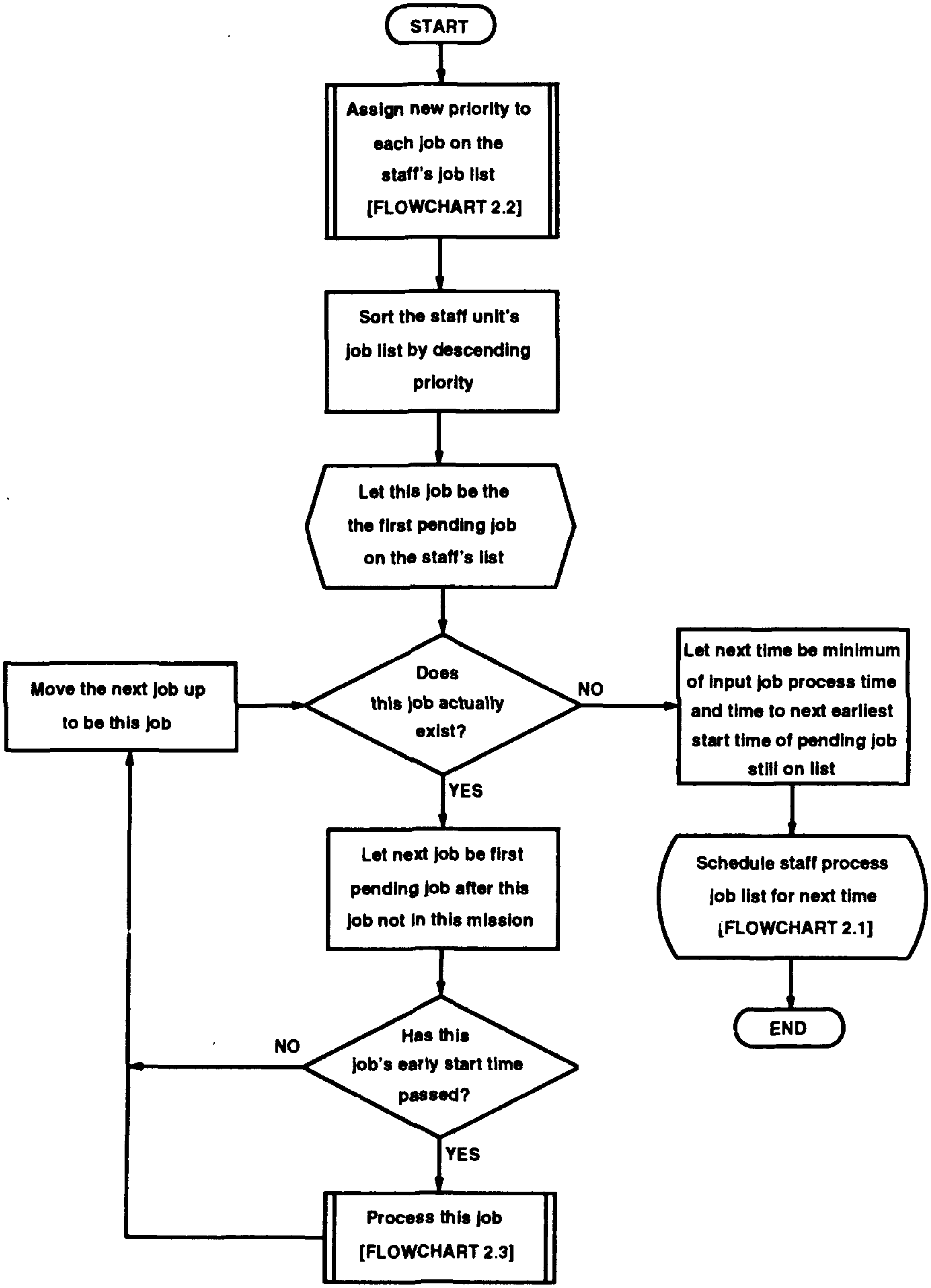
FLOWCHART 1.5 - DETERMINE ENGINEER STAFF'S CAPABILITY TO DO MISSION


START
Mark requestor as incapable until proven otherwise
Is this a mobility task or will requestor have site in tactical area?
NO
END
YES
For each technique for this side, task, and feature
NO
Has the last suitable technique been processed?
YES
Compare equipment and supplies required for technique with those available to unit
Does unit have all required equipment and supplies?
NO
YES
Is this technique best way (preferred, fastest, most effective)?
NO
YES
Choose this technique and mark requestor capable

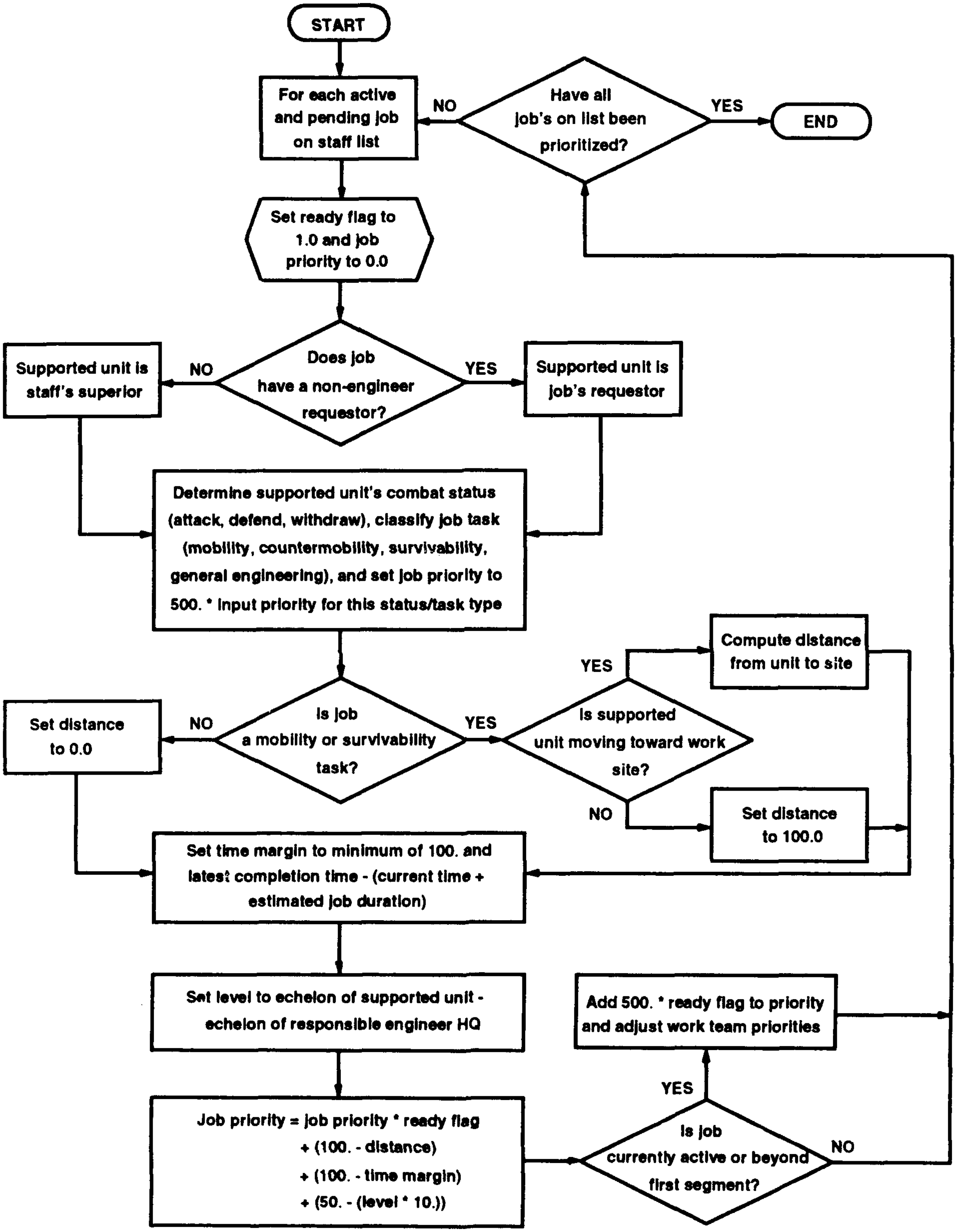
FLOWCHART 1.6 - DETERMINE REQUESTOR'S CAPABILITY TO PERFORM TASK IMPLICITLY


START
Let chosen staff be the current unit
Is current unit at technique's echelon or without qualifying subordinates?
YES
Assign mission to the current unit
END
NO
Change current unit to chosen subordinate if one qualified
YES
Has last subordinate of current unit been processed?
NO
For each engineer HQ subordinate to the current unit
Is subordinate authorized to receive the required supplies?
NO
YES
Is the work site in the subordinate's tactical area?
NO
YES
Determine subordinate's distance to work site and amount of equipment it has to do the work
Does subordinate have most equipment found thus far?
NO
Is equipment same but subordinate closer to work site?
NO
YES
YES
Choose this subordinate of the current unit

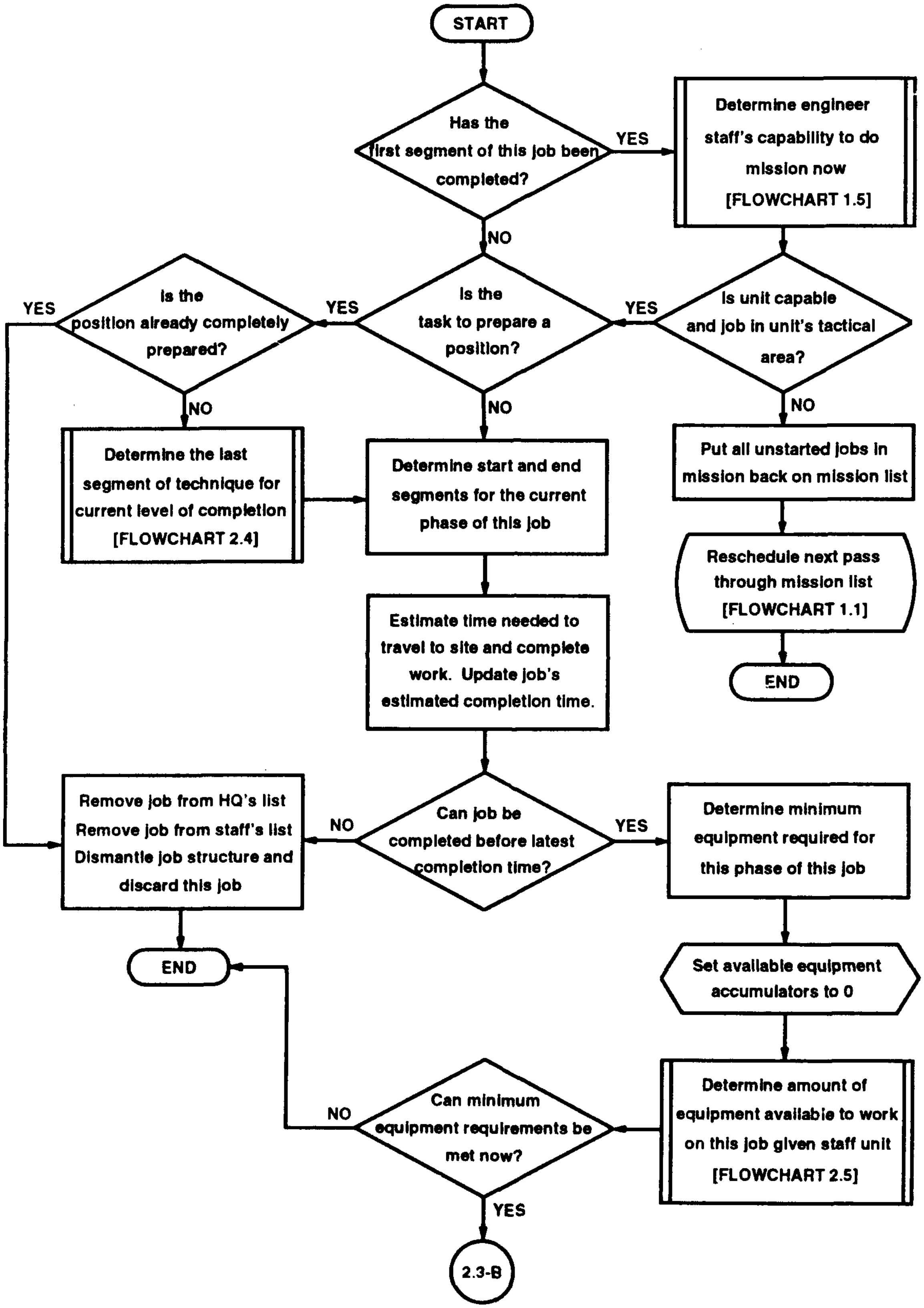
FLOWCHART 1.7 - FIND HQ SUBORDINATE TO STAFF TO ASSIGN MISSION


START
Assign new priority to each job on the staff's job list [FLOWCHART 2.2]
Sort the staff unit's job list by descending priority
Let this job be the the first pending job on the staff's list
Does this job actually exist?
NO
Let next time be minimum of input job process time and time to next earliest start time of pending job still on list
Schedule staff process job list for next time [FLOWCHART 2.1]
END
YES
Move the next job up to be this job
Let next job be first pending job after this job not in this mission
Has this job's early start time passed?
NO
YES
Process this job [FLOWCHART 2.3]

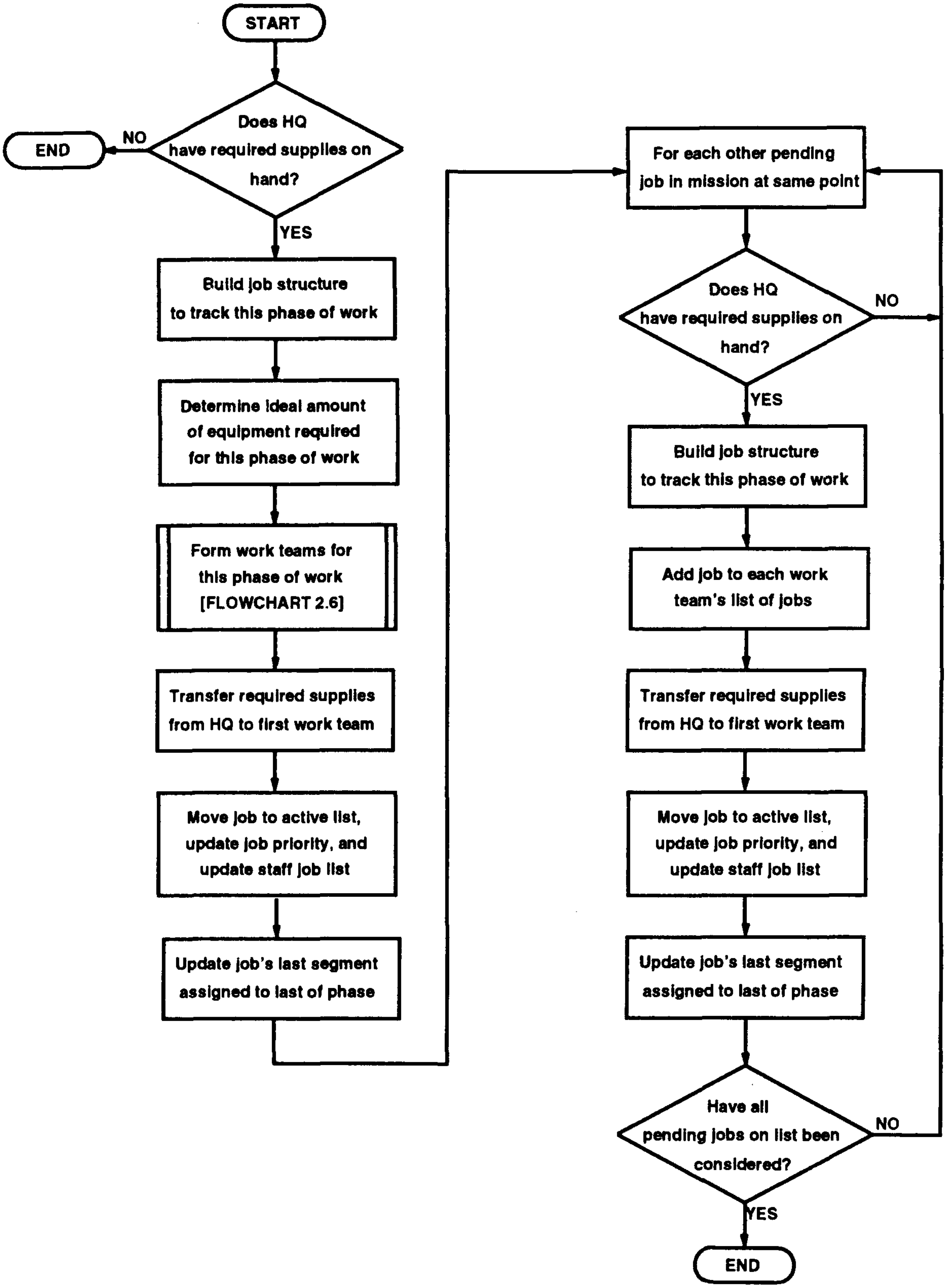
FLOWCHART 2.1 - STAFF UNIT PROCESSES COMMAND'S JOB LIST


START
For each active and pending job on staff list
NO
Have all job's on list been prioritized?
YES
END
Set ready flag to 1.0 and job priority to 0.0
Supported unit is staff's superior
NO
Does job have a non-engineer requestor?
YES
Supported unit is job's requestor
Determine supported unit's combat status (attack, defend, withdraw), classify job task (mobility, countermobility, survivability, general engineering), and set job priority to 500. * input priority for this status/task type
Compute distance from unit to site
YES
Set distance to 0.0
NO
Is job a mobility or survivability task?
YES
Is supported unit moving toward work site?
NO
Set distance to 100.0
Set time margin to minimum of 100. and latest completion time - (current time + estimated job duration)
Set level to echelon of supported unit - echelon of responsible engineer HQ
Add 500. * ready flag to priority and adjust work team priorities
YES
Job priority = job priority * ready flag
+ (100. - distance)
+ (100. - time margin)
+ (50. - (level * 10.))
Is job currently active or beyond first segment?
NO

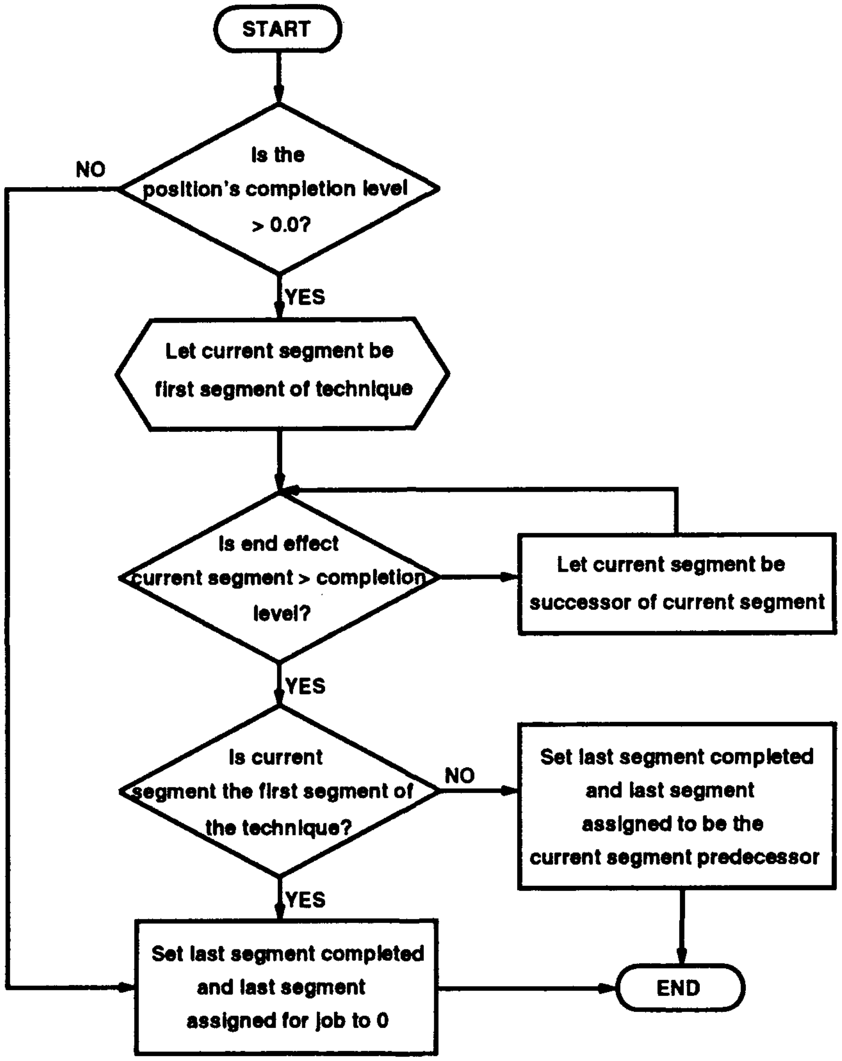
FLOWCHART 2.2 - ASSIGN PRIORITY TO EACH JOB ON STAFF'S JOB LIST


START
Has the first segment of this job been completed?
YES
Determine engineer staff's capability to do mission now [FLOWCHART 1.5]
NO
Is the position already completely prepared?
YES
Is the task to prepare a position?
YES
Is unit capable and job in unit's tactical area?
YES
NO
NO
NO
Determine the last segment of technique for current level of completion [FLOWCHART 2.4]
Determine start and end segments for the current phase of this job
Put all unstarted jobs in mission back on mission list
Reschedule next pass through mission list [FLOWCHART 1.1]
END
Estimate time needed to travel to site and complete work. Update job's estimated completion time.
Remove job from HQ's list
Remove job from staff's list
Dismantle job structure and discard this job
NO
Can job be completed before latest completion time?
YES
Determine minimum equipment required for this phase of this job
END
Set available equipment accumulators to 0
NO
Can minimum equipment requirements be met now?
Determine amount of equipment available to work on this job given staff unit [FLOWCHART 2.5]
YES
2.3-B

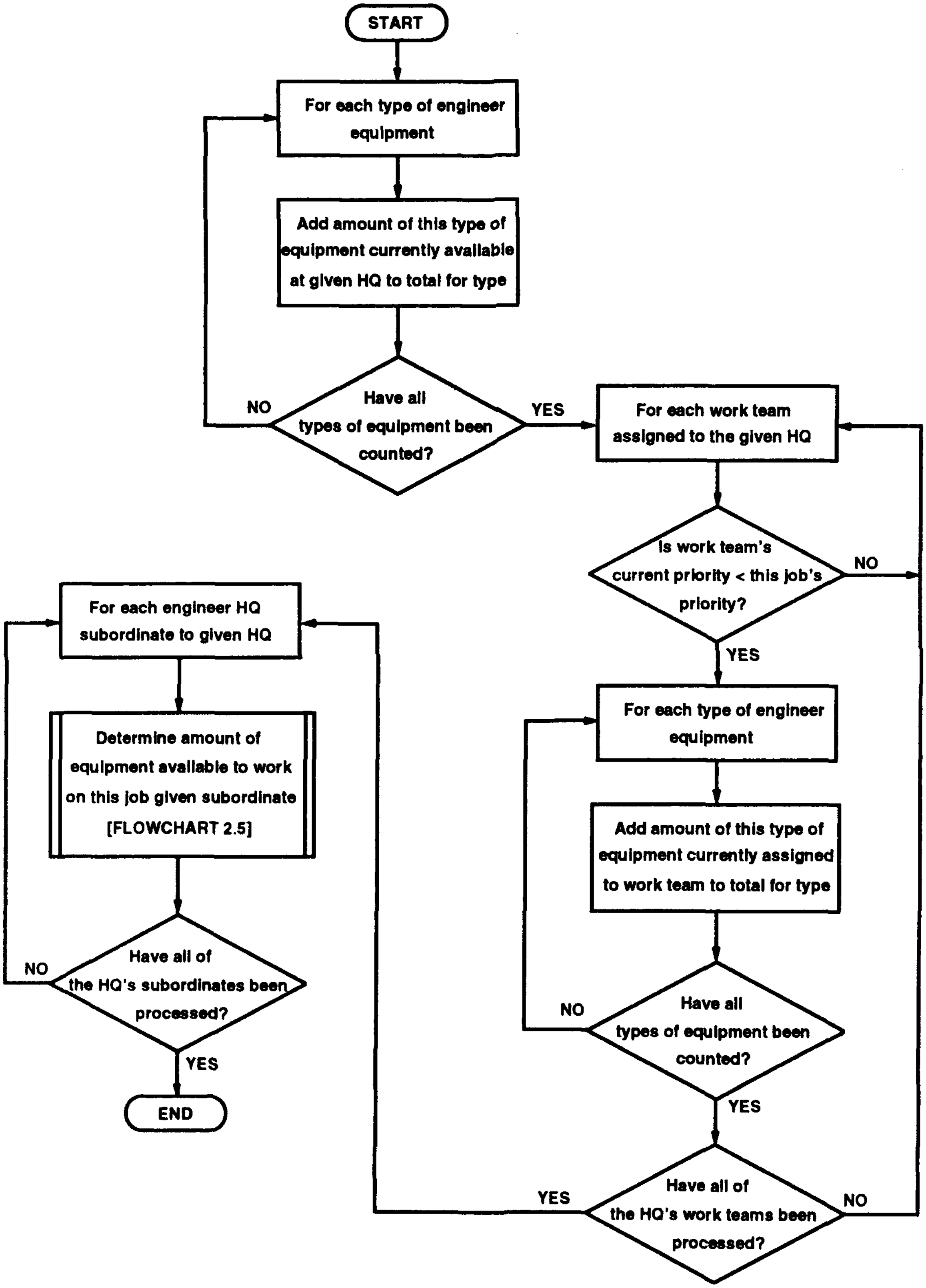
FLOWCHART 2.3 - PROCESS THIS JOB


START
Does HQ have required supplies on hand?
NO
END
YES
Build job structure to track this phase of work
Determine ideal amount of equipment required for this phase of work
Form work teams for this phase of work [FLOWCHART 2.6]
Transfer required supplies from HQ to first work team
Move job to active list, update job priority, and update staff job list
Update job's last segment assigned to last of phase
For each other pending job in mission at same point
Does HQ have required supplies on hand?
NO
YES
Build job structure to track this phase of work
Add job to each work team's list of jobs
Transfer required supplies from HQ to first work team
Move job to active list, update job priority, and update staff job list
Update job's last segment assigned to last of phase
Have all pending jobs on list been considered?
NO
YES
END

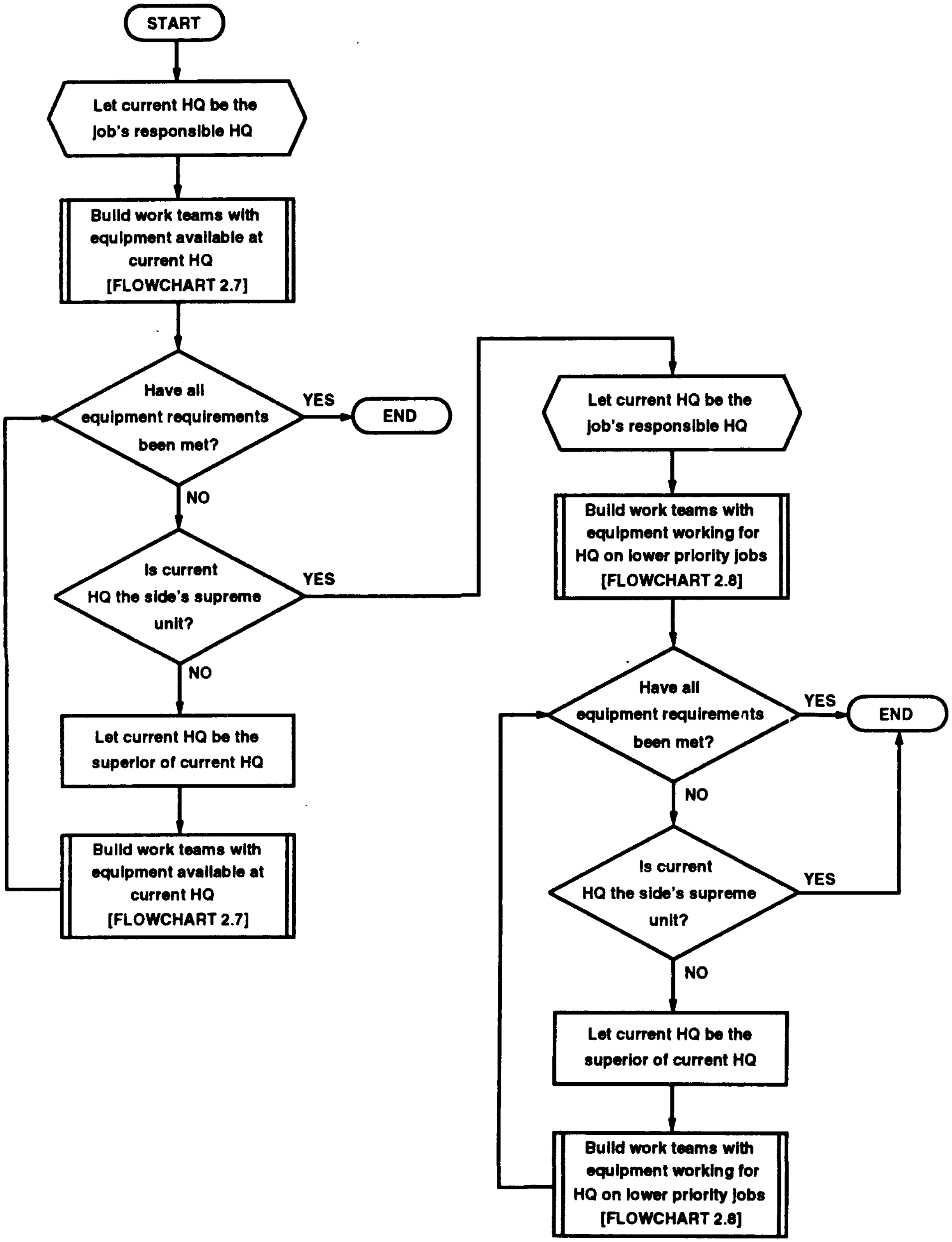
FLOWCHART 2.3-B - PROCESS THIS JOB (CONTINUED)


START
Is the position's completion level > 0.0?
NO
YES
Let current segment be first segment of technique
Is end effect current segment > completion level?
Let current segment be successor of current segment
YES
Is current segment the first segment of the technique?
NO
Set last segment completed and last segment assigned to be the current segment predecessor
YES
Set last segment completed and last segment assigned for job to 0
END

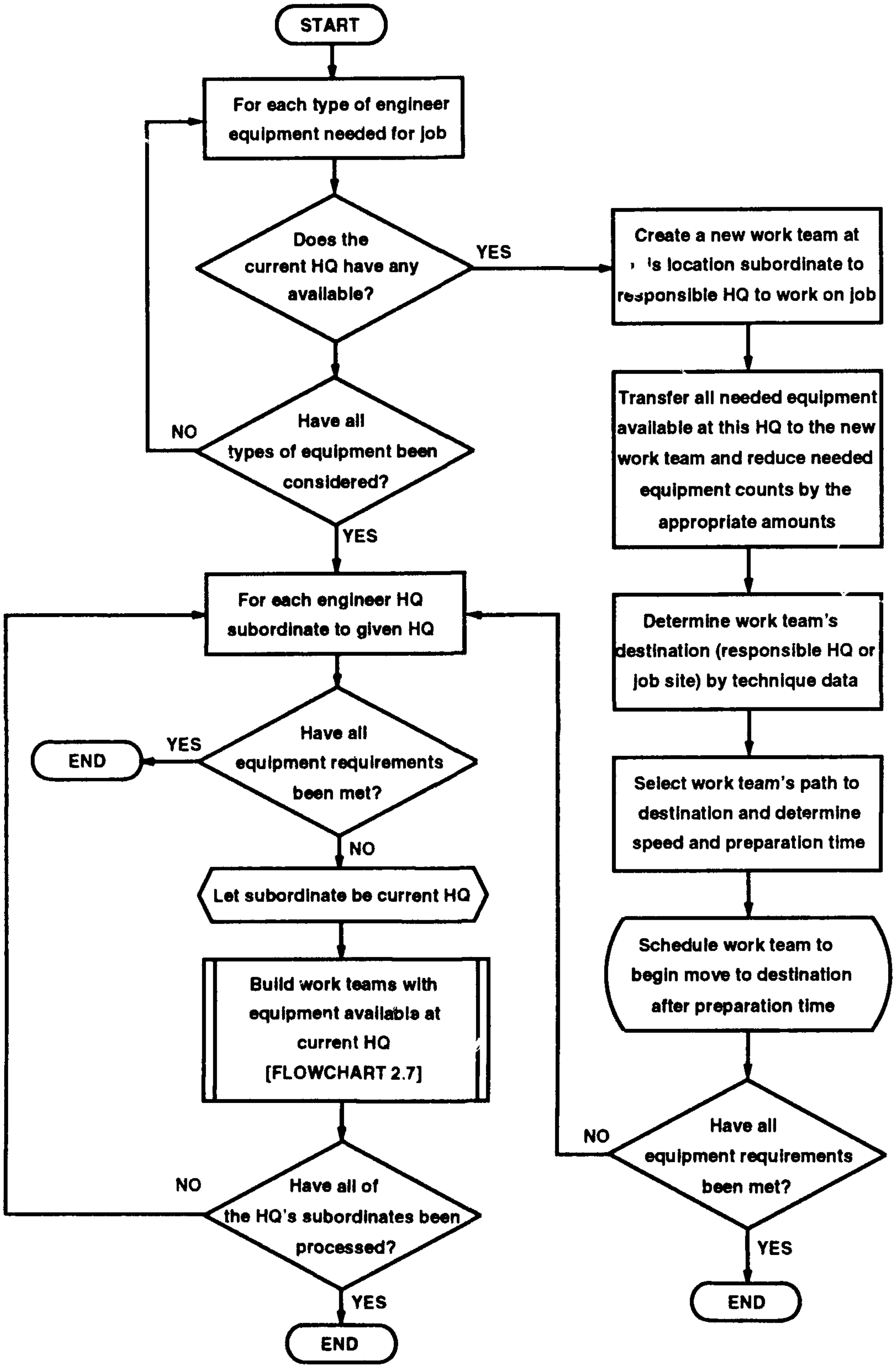
FLOWCHART 2.4 - DETERMINE LAST SEGMENT COMPLETED FOR SURVIVABILITY TASK


START
For each type of engineer equipment
Add amount of this type of equipment currently available at given HQ to total for type
Have all types of equipment been counted?
NO
YES
For each work team assigned to the given HQ
Is work team's current priority < this job's priority?
NO
YES
For each type of engineer equipment
Add amount of this type of equipment currently assigned to work team to total for type
Have all types of equipment been counted?
NO
YES
Have all of the HQ's work teams been processed?
YES
NO
For each engineer HQ subordinate to given HQ
Determine amount of equipment available to work on this job given subordinate [FLOWCHART 2.5]
Have all of the HQ's subordinates been processed?
NO
YES
END

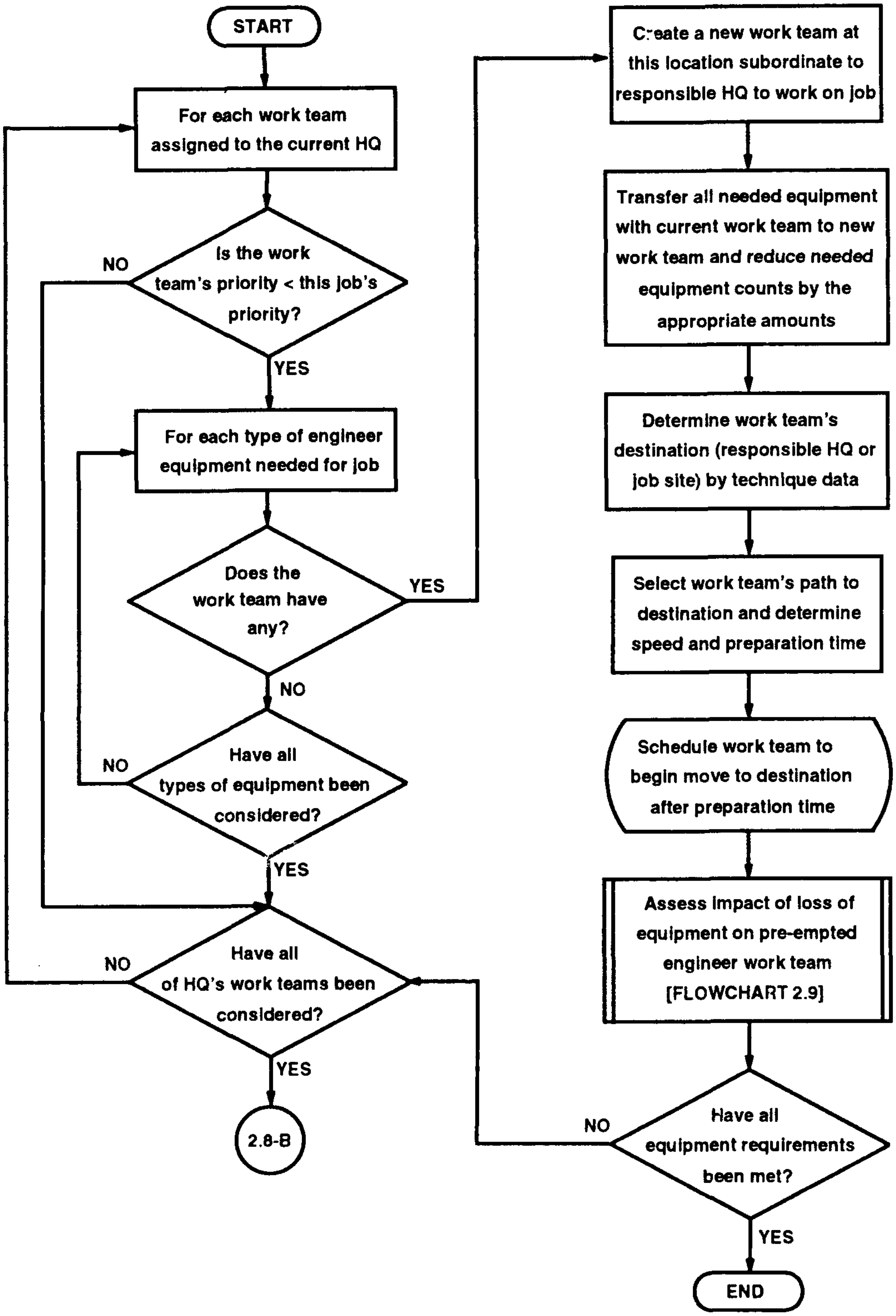
FLOWCHART 2.5 - DETERMINE AMOUNT OF EQUIPMENT AVAILABLE FOR A JOB


START
Let current HQ be the job's responsible HQ
Build work teams with equipment available at current HQ [FLOWCHART 2.7]
Have all equipment requirements been met?
YES
END
NO
Is current HQ the side's supreme unit?
YES
NO
Let current HQ be the superior of current HQ
Build work teams with equipment available at current HQ [FLOWCHART 2.7]
Let current HQ be the job's responsible HQ
Build work teams with equipment working for HQ on lower priority jobs [FLOWCHART 2.8]
Have all equipment requirements been met?
YES
END
NO
Is current HQ the side's supreme unit?
YES
NO
Let current HQ be the superior of current HQ
Build work teams with equipment working for HQ on lower priority jobs [FLOWCHART 2.8]

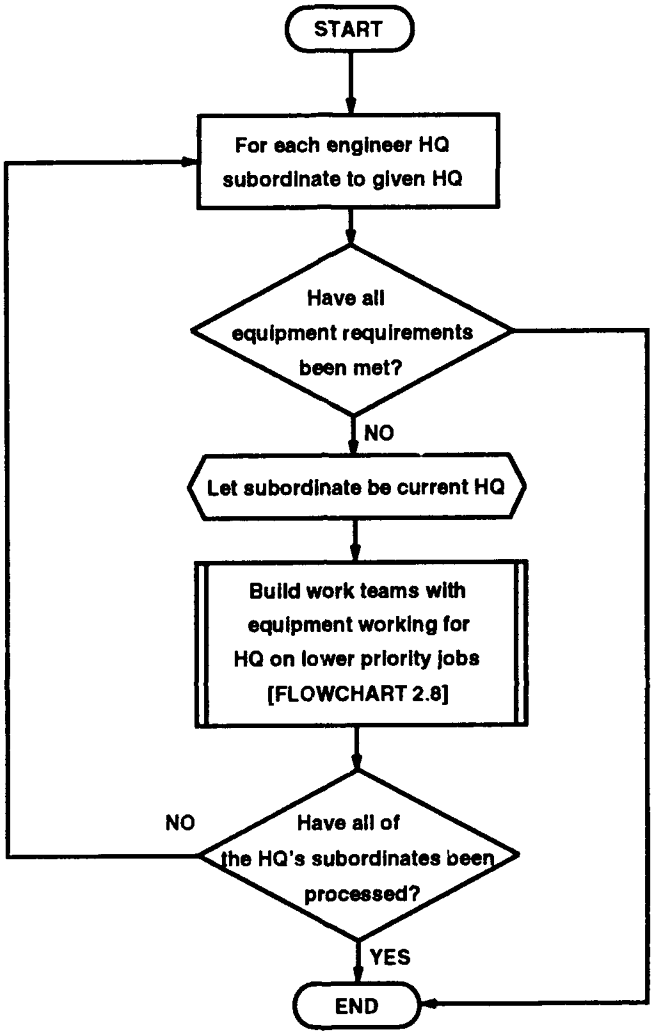
FLOWCHART 2.6 - FORM WORK TEAMS FOR THIS PHASE OF WORK


START
For each type of engineer equipment needed for job
Does the current HQ have any available?
YES
Create a new work team at › 's location subordinate to responsible HQ to work on job
Transfer all needed equipment available at this HQ to the new work team and reduce needed equipment counts by the appropriate amounts
Determine work team's destination (responsible HQ or job site) by technique data
Select work team's path to destination and determine speed and preparation time
Schedule work team to begin move to destination after preparation time
Have all equipment requirements been met?
NO
YES
END
Have all types of equipment been considered?
NO
YES
For each engineer HQ subordinate to given HQ
Have all equipment requirements been met?
YES
END
NO
Let subordinate be current HQ
Build work teams with equipment available at current HQ [FLOWCHART 2.7]
Have all of the HQ's subordinates been processed?
NO
YES
END

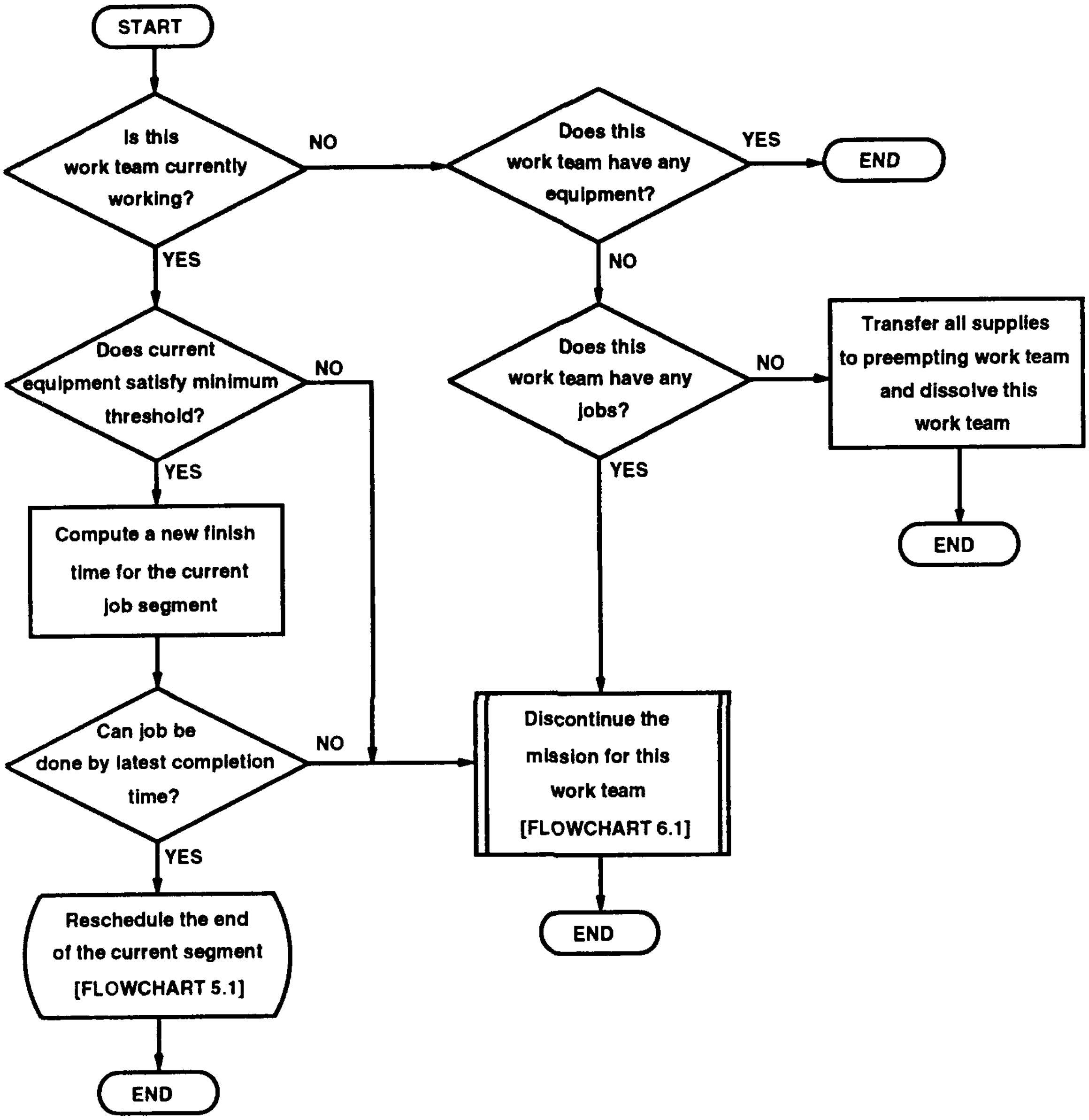
FLOWCHART 2.7 - BUILD WORK TEAM AT CURRENT HQ


START
For each work team assigned to the current HQ
Is the work team's priority < this job's priority?
NO
YES
For each type of engineer equipment needed for job
Does the work team have any?
YES
NO
Have all types of equipment been considered?
NO
YES
Have all of HQ's work teams been considered?
NO
YES
2.8-B
C:eate a new work team at this location subordinate to responsible HQ to work on job
Transfer all needed equipment with current work team to new work team and reduce needed equipment counts by the appropriate amounts
Determine work team's destination (responsible HQ or job site) by technique data
Select work team's path to destination and determine speed and preparation time
Schedule work team to begin move to destination after preparation time
Assess impact of loss of equipment on pre-empted engineer work team [FLOWCHART 2.9]
Have all equipment requirements been met?
NO
YES
END

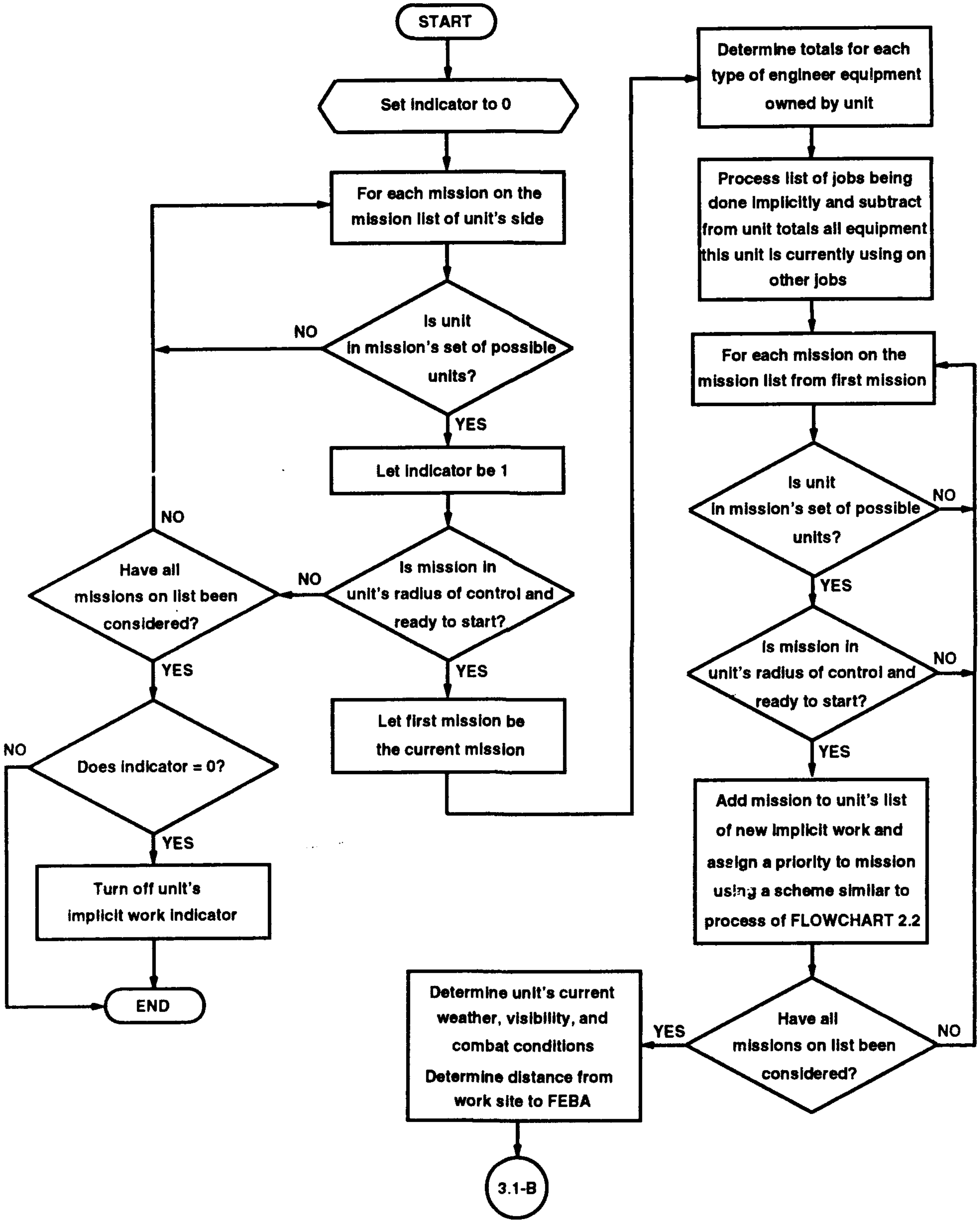
FLOWCHART 2.8 - BUILD WORK TEAMS WITH EQUIPMENT WORKING FOR HQ


START
For each engineer HQ subordinate to given HQ
Have all equipment requirements been met?
NO
Let subordinate be current HQ
Build work teams with equipment working for HQ on lower priority jobs [FLOWCHART 2.8]
NO
Have all of the HQ's subordinates been processed?
YES
END

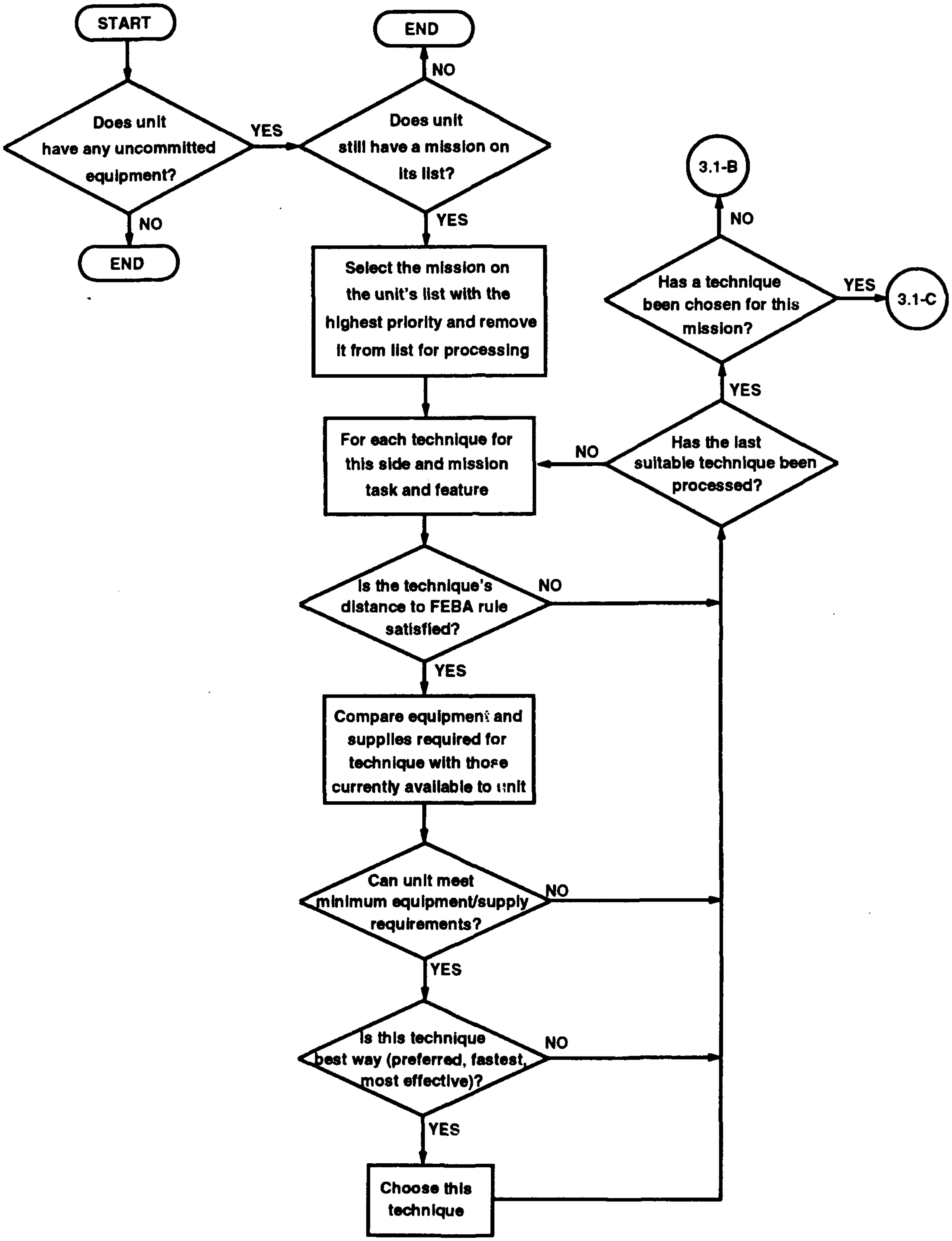
FLOWCHART 2.8-B - BUILD WORK TEAMS WITH EQUIPMENT WORKING FOR HQ (CONTINUED)


START
Is this work team currently working?
NO
YES
Does this work team have any equipment?
YES
END
NO
Does current equipment satisfy minimum threshold?
NO
YES
Does this work team have any jobs?
NO
Transfer all supplies to preempting work team and dissolve this work team
END
YES
Compute a new finish time for the current job segment
Can job be done by latest completion time?
NO
Discontinue the mission for this work team [FLOWCHART 6.1]
YES
END
Reschedule the end of the current segment [FLOWCHART 5.1]
END

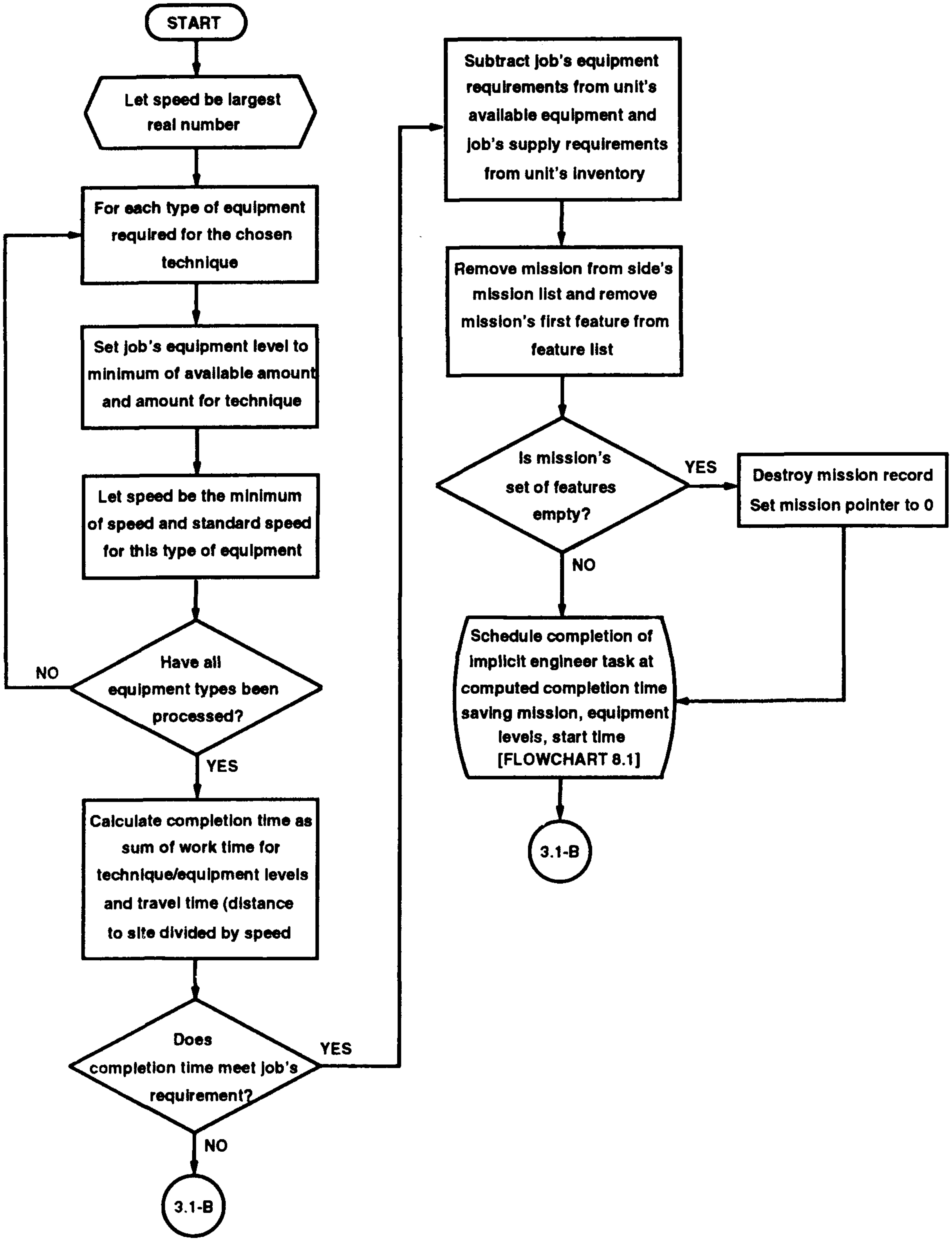
FLOWCHART 2.9 - ASSESS IMPACT OF LOSS OF EQUIPMENT ON PREEMPTED WORK TEAM


START
Set indicator to 0
For each mission on the mission list of unit's side
Is unit in mission's set of possible units?
NO
YES
Let indicator be 1
Is mission in unit's radius of control and ready to start?
NO
YES
Have all missions on list been considered?
NO
YES
Let first mission be the current mission
Does indicator = 0?
NO
YES
Turn off unit's implicit work indicator
END
Determine totals for each type of engineer equipment owned by unit
Process list of jobs being done implicitly and subtract from unit totals all equipment this unit is currently using on other jobs
For each mission on the mission list from first mission
Is unit in mission's set of possible units?
NO
YES
Is mission in unit's radius of control and ready to start?
NO
YES
Add mission to unit's list of new implicit work and assign a priority to mission using a scheme similar to process of FLOWCHART 2.2
Have all missions on list been considered?
NO
YES
Determine unit's current weather, visibility, and combat conditions
Determine distance from work site to FEBA
3.1-B

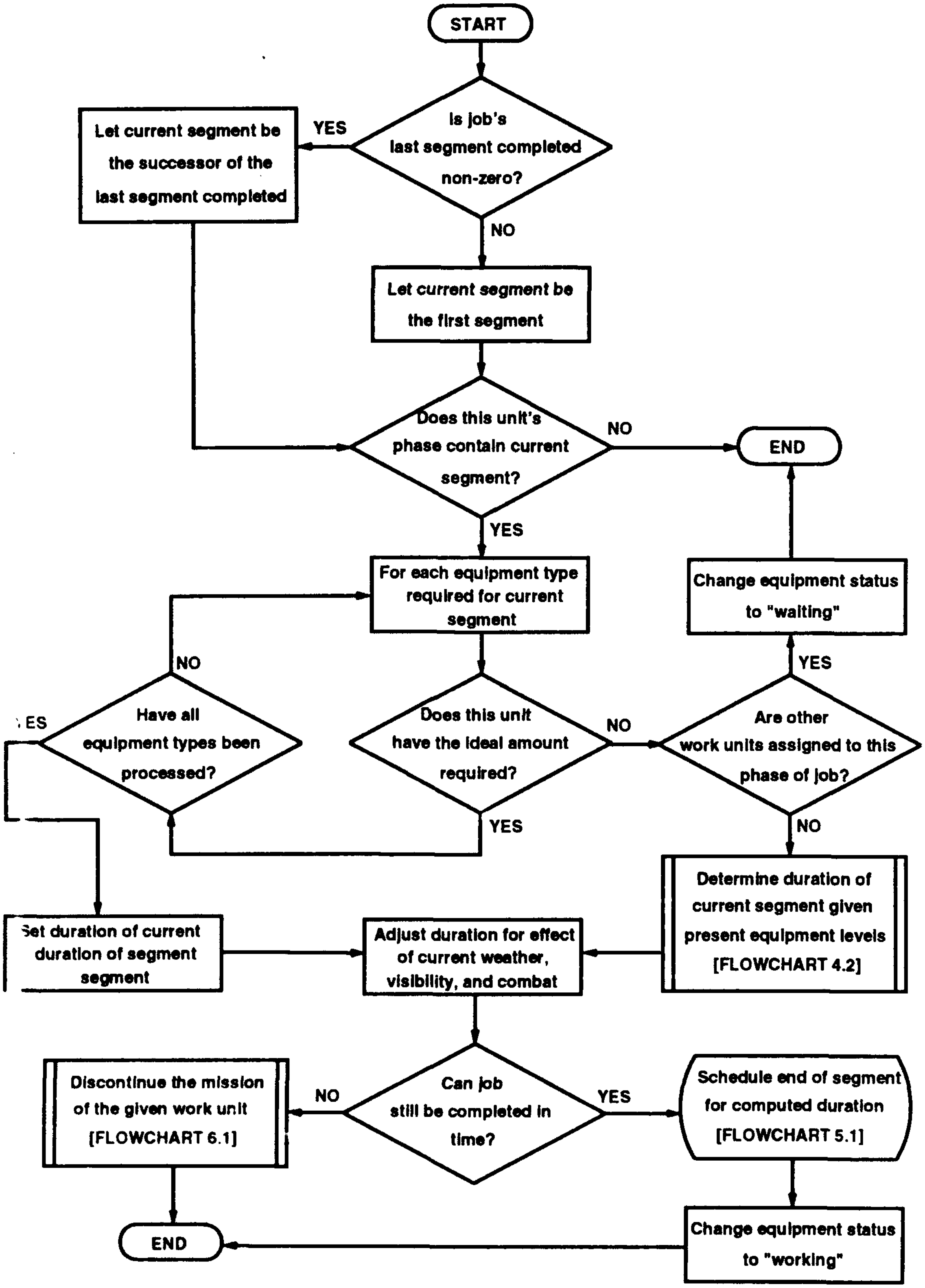
FLOWCHART 3.1 - PERFORM WORK IMPLICITLY


START
Does unit have any uncommitted equipment?
YES
NO
END
Does unit still have a mission on its list?
NO
END
YES
Select the mission on the unit's list with the highest priority and remove it from list for processing
For each technique for this side and mission task and feature
Is the technique's distance to FEBA rule satisfied?
NO
YES
Compare equipment and supplies required for technique with those currently available to unit
Can unit meet minimum equipment/supply requirements?
NO
YES
Is this technique best way (preferred, fastest, most effective)?
NO
YES
Choose this technique
Has the last suitable technique been processed?
NO
YES
Has a technique been chosen for this mission?
NO
3.1-B
YES
3.1-C

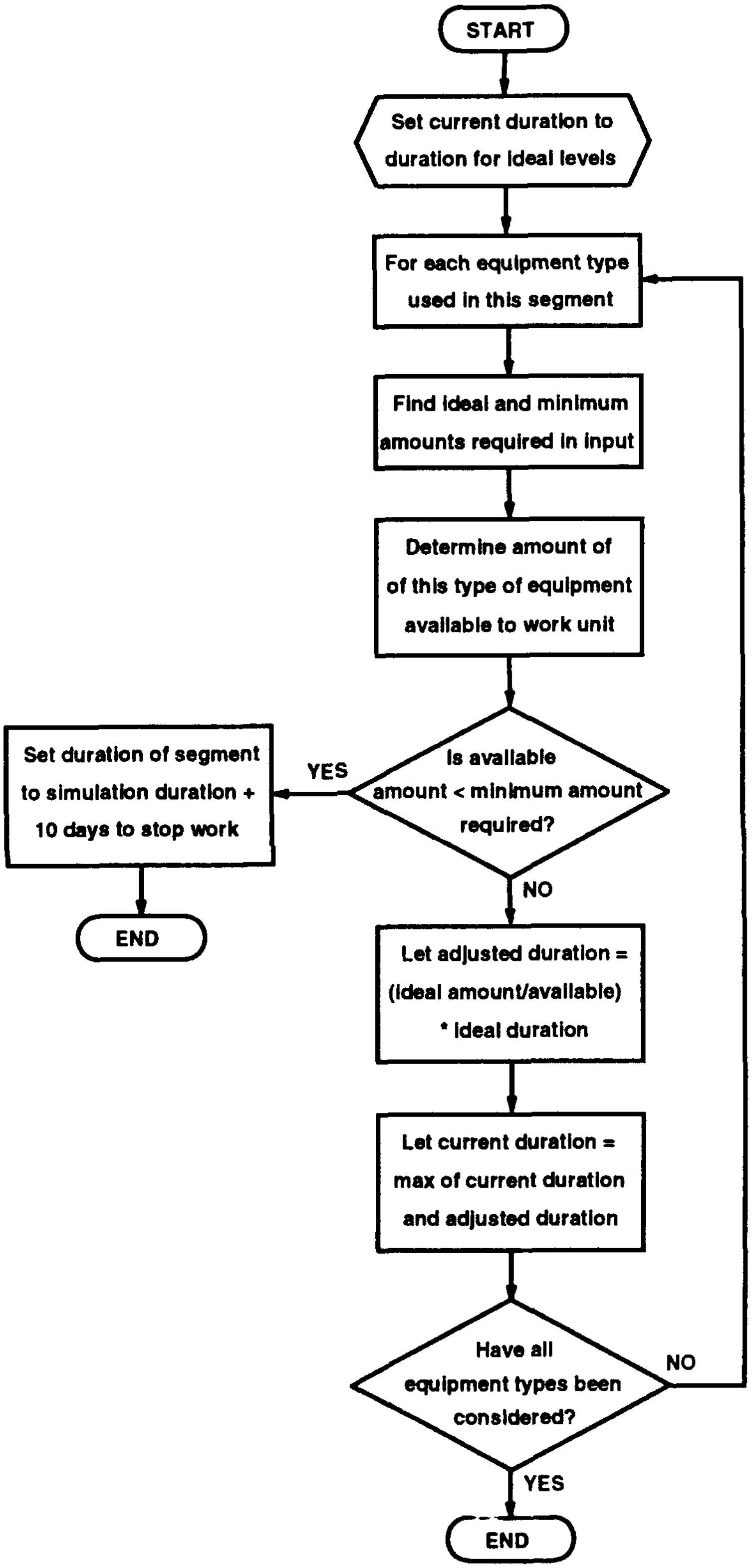
FLOWCHART 3.1-B - PERFORM WORK IMPLICITLY (CONTINUED)


START
Let speed be largest real number
For each type of equipment required for the chosen technique
Set job's equipment level to minimum of available amount and amount for technique
Let speed be the minimum of speed and standard speed for this type of equipment
Have all equipment types been processed?
NO
YES
Calculate completion time as sum of work time for technique/equipment levels and travel time (distance to site divided by speed
Does completion time meet job's requirement?
YES
NO
3.1-B
Subtract job's equipment requirements from unit's available equipment and job's supply requirements from unit's inventory
Remove mission from side's mission list and remove mission's first feature from feature list
Is mission's set of features empty?
YES
Destroy mission record Set mission pointer to 0
NO
Schedule completion of implicit engineer task at computed completion time saving mission, equipment levels, start time [FLOWCHART 8.1]
3.1-B

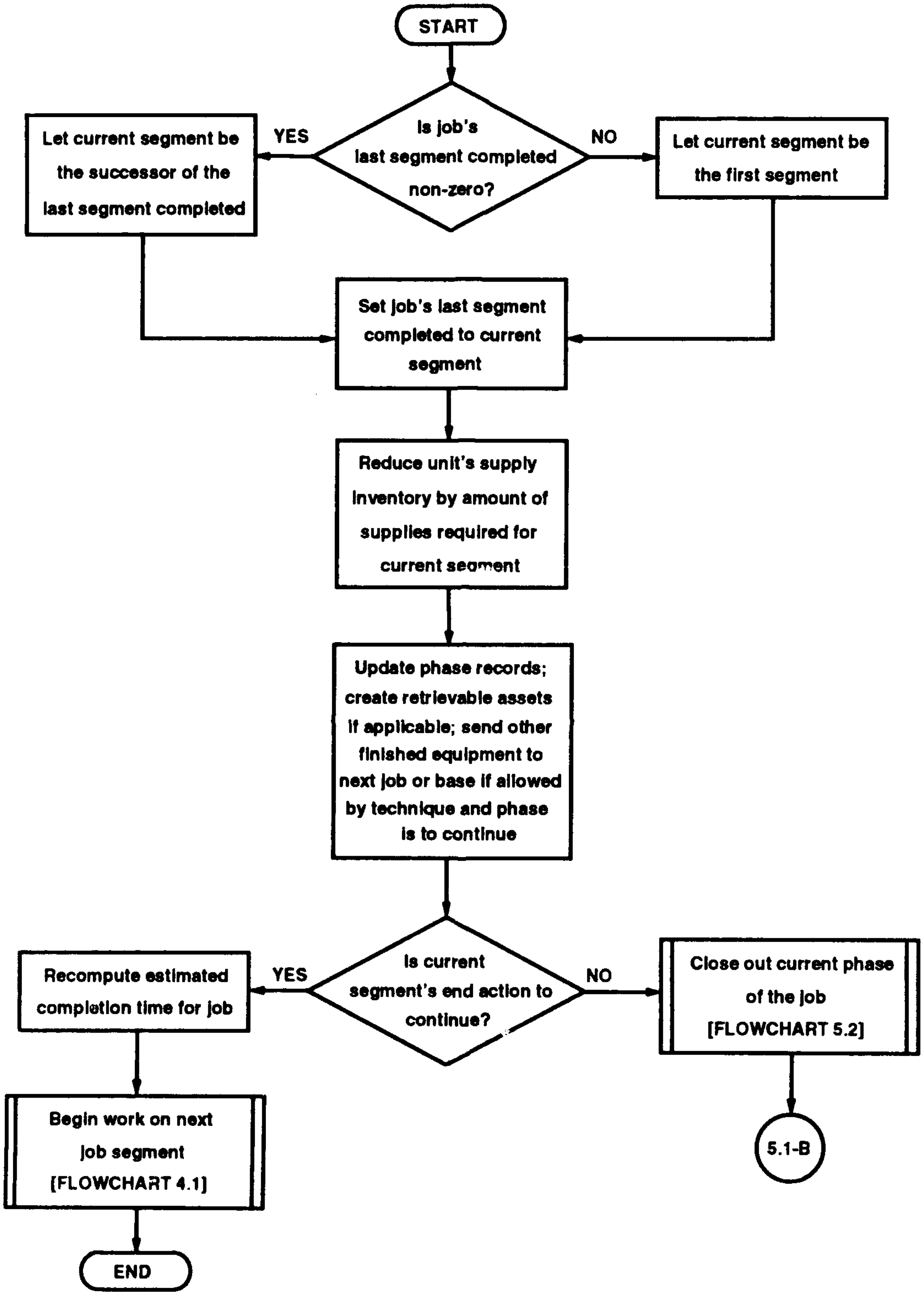
FLOWCHART 3.1-C - PERFORM WORK IMPLICITLY (CONTINUED)


START
Is job's last segment completed non-zero?
YES
Let current segment be the successor of the last segment completed
NO
Let current segment be the first segment
Does this unit's phase contain current segment?
NO
END
YES
For each equipment type required for current segment
Change equipment status to "waiting"
NO
YES
Have all equipment types been processed?
ES
Does this unit have the ideal amount required?
NO
Are other work units assigned to this phase of job?
YES
NO
Determine duration of current segment given present equipment levels [FLOWCHART 4.2]
Set duration of current duration of segment segment
Adjust duration for effect of current weather, visibility, and combat
Discontinue the mission of the given work unit [FLOWCHART 6.1]
NO
Can job still be completed in time?
YES
Schedule end of segment for computed duration [FLOWCHART 5.1]
END
Change equipment status to "working"

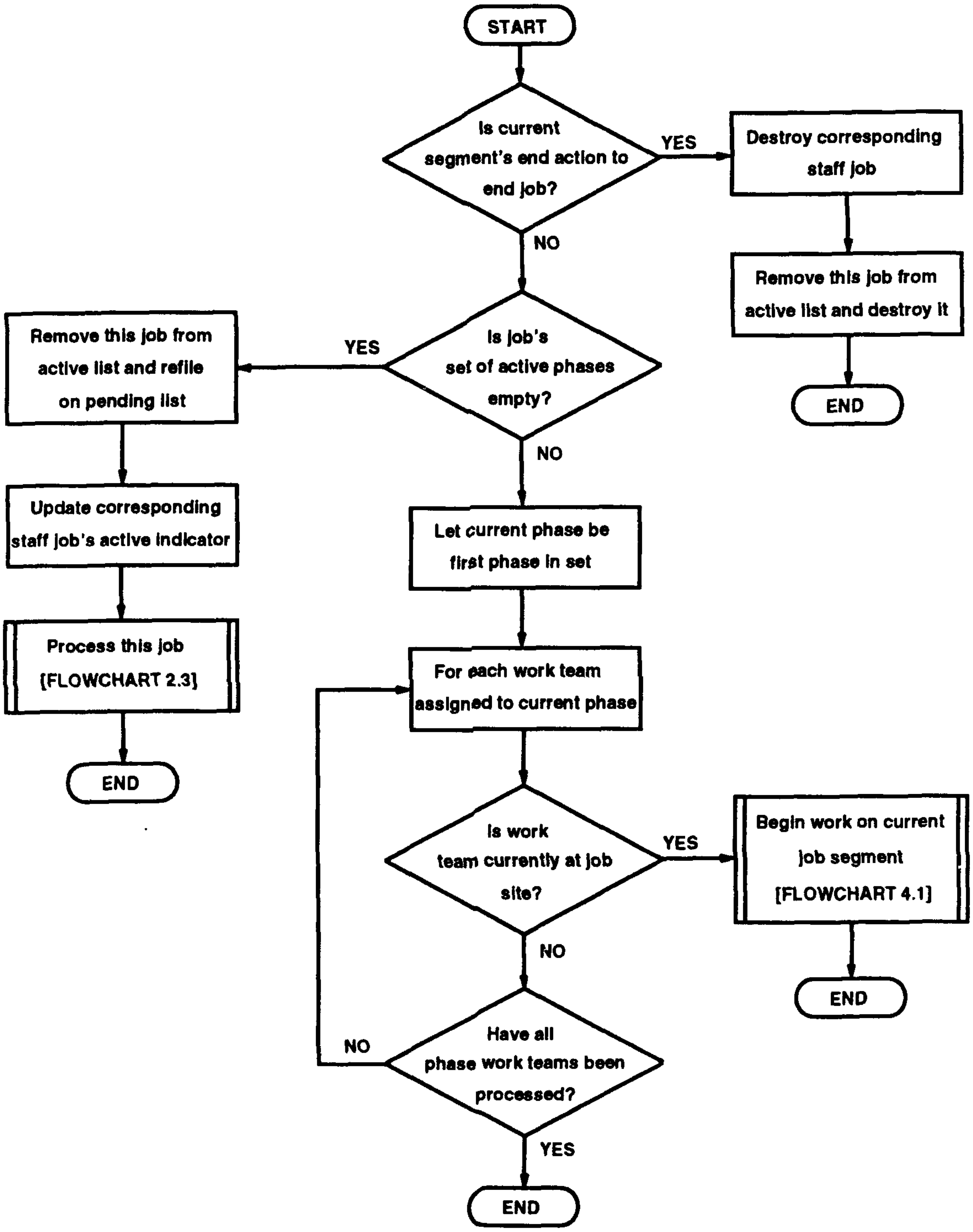
FLOWCHART 4.1 - BEGIN WORK ON CURRENT JOB SEGMENT


START
Set current duration to duration for ideal levels
For each equipment type used in this segment
Find ideal and minimum amounts required in input
Determine amount of of this type of equipment available to work unit
Is available amount < minimum amount required?
YES
Set duration of segment to simulation duration + 10 days to stop work
END
NO
Let adjusted duration = (ideal amount/available) * ideal duration
Let current duration = max of current duration and adjusted duration
Have all equipment types been considered?
NO
YES
END

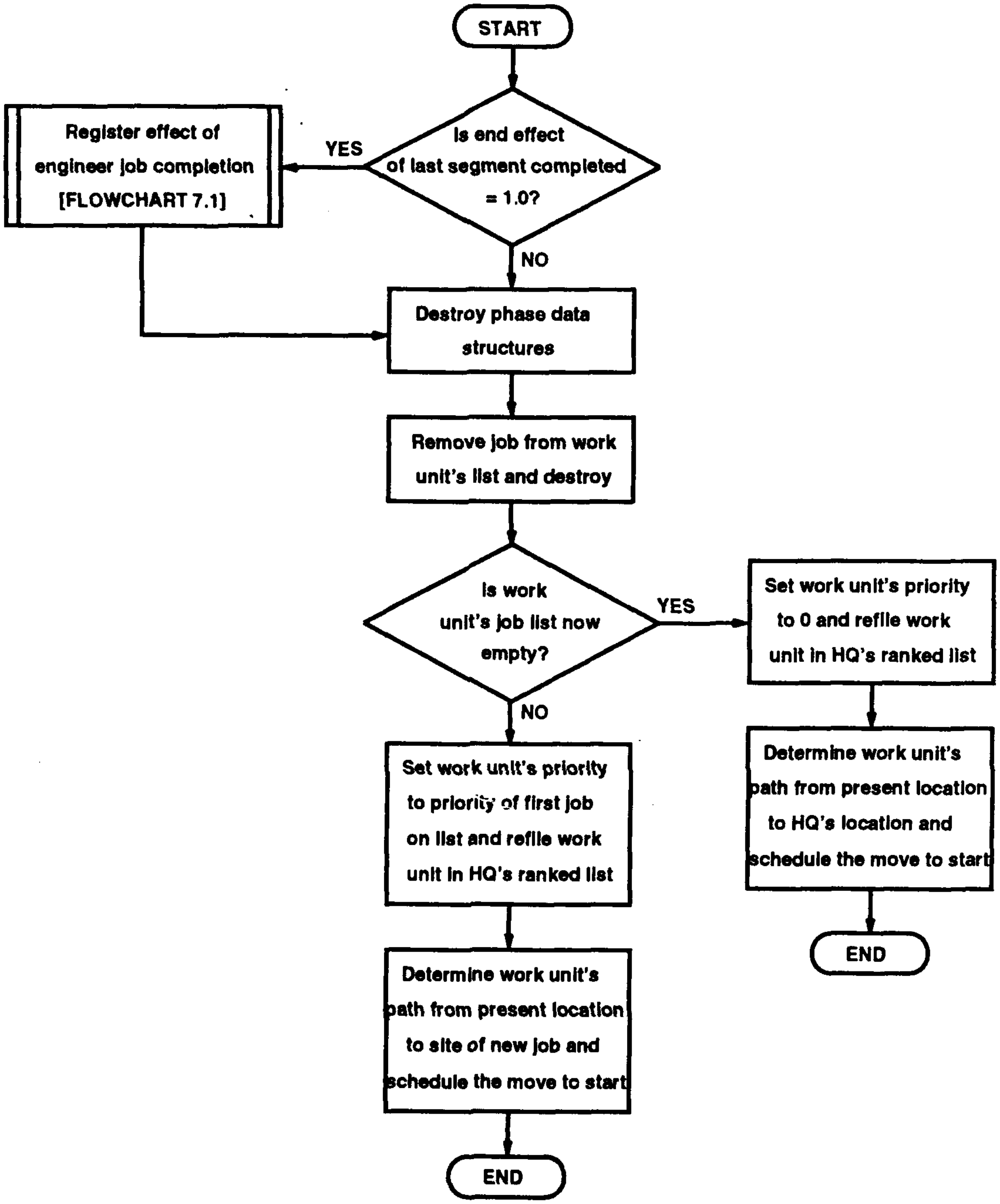
FLOWCHART 4.2 - DETERMINE DURATION OF CURRENT SEGMENT GIVEN EQUIPMENT LEVELS


START
Is job's last segment completed non-zero?
YES
Let current segment be the successor of the last segment completed
NO
Let current segment be the first segment
Set job's last segment completed to current segment
Reduce unit's supply inventory by amount of supplies required for current seoment
Update phase records; create retrievable assets if applicable; send other finished equipment to next job or base if allowed by technique and phase is to continue
Is current segment's end action to continue?
YES
Recompute estimated completion time for job
Begin work on next job segment [FLOWCHART 4.1]
END
NO
Close out current phase of the job [FLOWCHART 5.2]
5.1-B

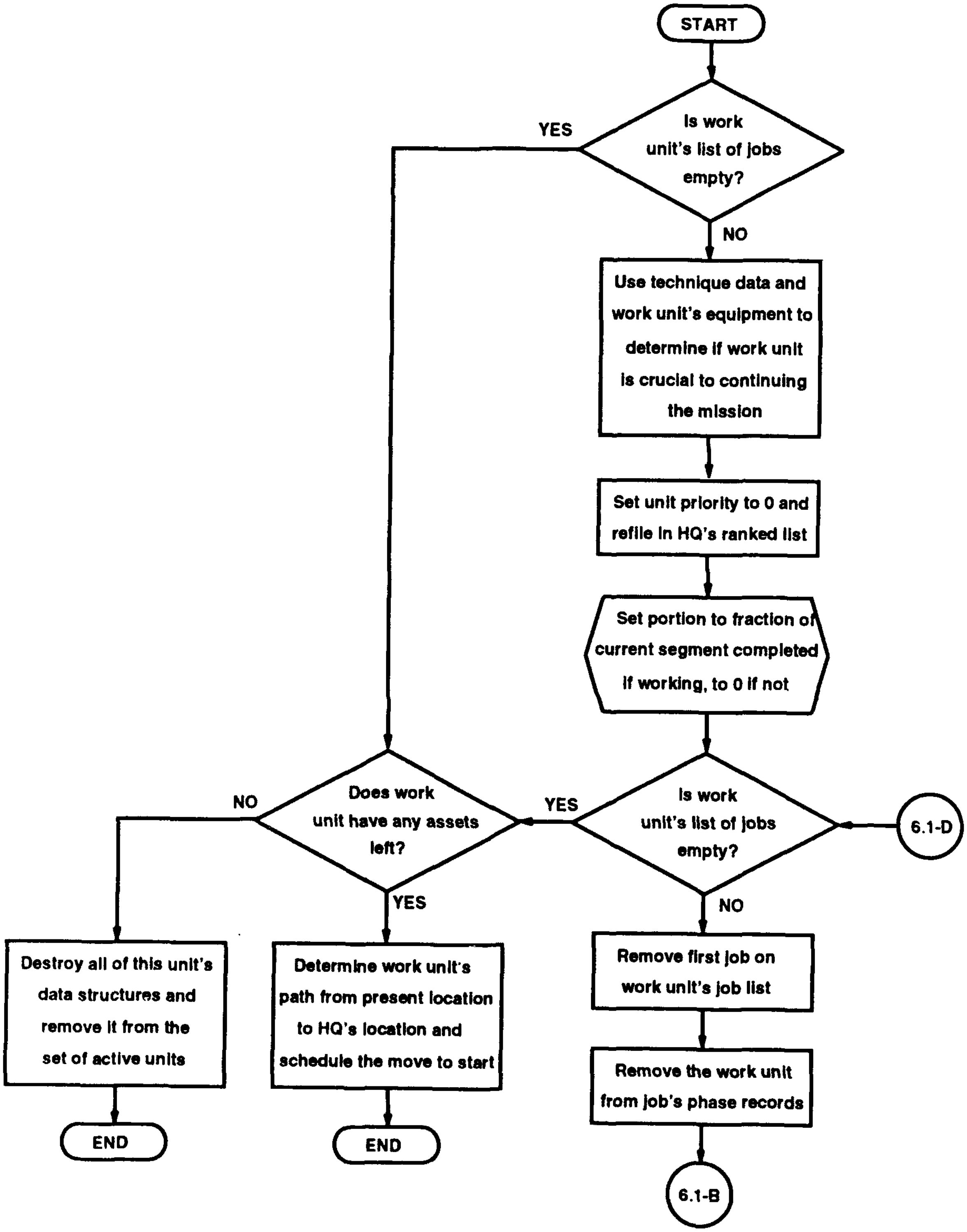
FLOWCHART 5.1 - END OF JOB SEGMENT


START
Is current segment's end action to end job?
YES
Destroy corresponding staff job
Remove this job from active list and destroy it
END
NO
Is job's set of active phases empty?
YES
Remove this job from active list and refile on pending list
Update corresponding staff job's active indicator
Process this job [FLOWCHART 2.3]
END
NO
Let current phase be first phase in set
For each work team assigned to current phase
Is work team currently at job site?
YES
Begin work on current job segment [FLOWCHART 4.1]
END
NO
Have all phase work teams been processed?
NO
YES
END

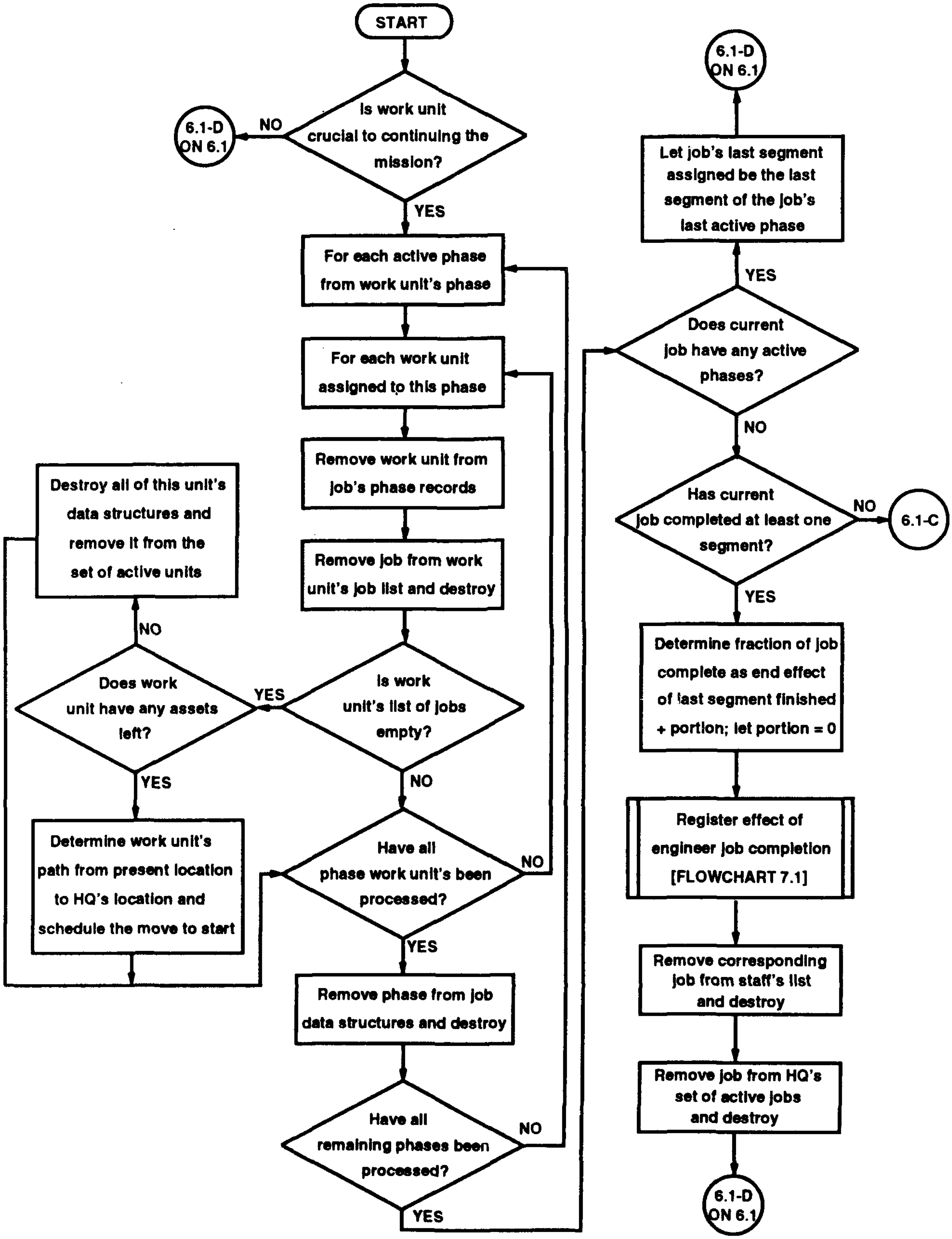
FLOWCHART 5.1-B - END OF JOB SEGMENT (CONTINUED)


START
Is end effect of last segment completed = 1.0?
YES
Register effect of engineer job completion [FLOWCHART 7.1]
NO
Destroy phase data structures
Remove job from work unit's list and destroy
Is work unit's job list now empty?
YES
Set work unit's priority to 0 and refile work unit in HQ's ranked list
Determine work unit's path from present location to HQ's location and schedule the move to start
END
NO
Set work unit's priority to priority of first job on list and refile work unit in HQ's ranked list
Determine work unit's path from present location to site of new job and schedule the move to start
END

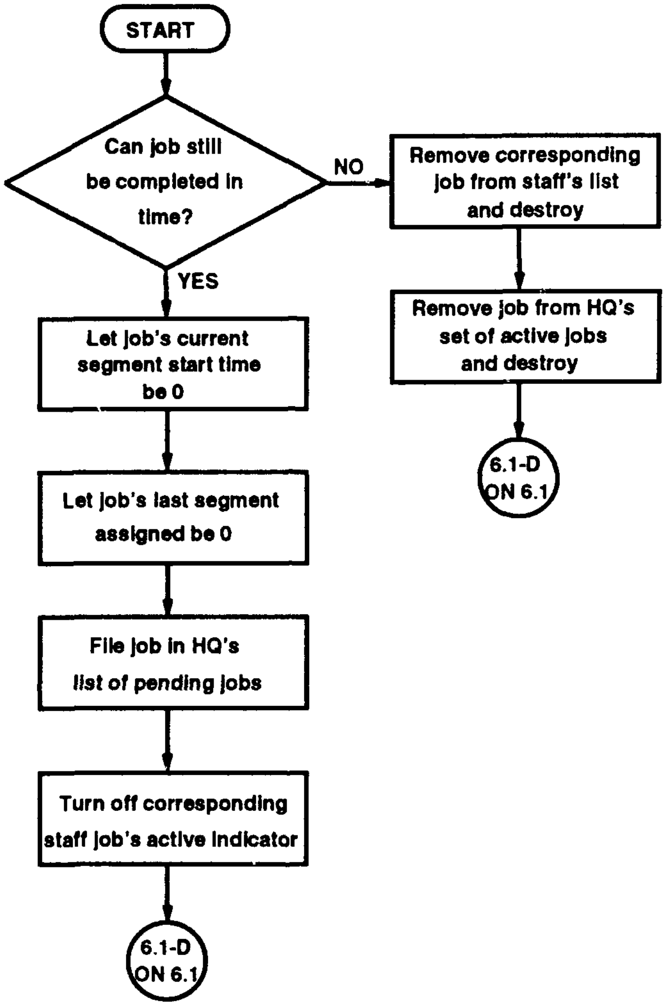
FLOWCHART 5.2 - CLOSE OUT CURRENT PHASE OF THIS JOB


START
Is work unit's list of jobs empty?
YES
NO
Use technique data and work unit's equipment to determine if work unit is crucial to continuing the mission
Set unit priority to 0 and refile in HQ's ranked list
Set portion to fraction of current segment completed if working, to 0 if not
Does work unit have any assets left?
NO
YES
Is work unit's list of jobs empty?
YES
NO
6.1-D
Destroy all of this unit's data structures and remove it from the set of active units
END
Determine work unit's path from present location to HQ's location and schedule the move to start
END
Remove first job on work unit's job list
Remove the work unit from job's phase records
6.1-B

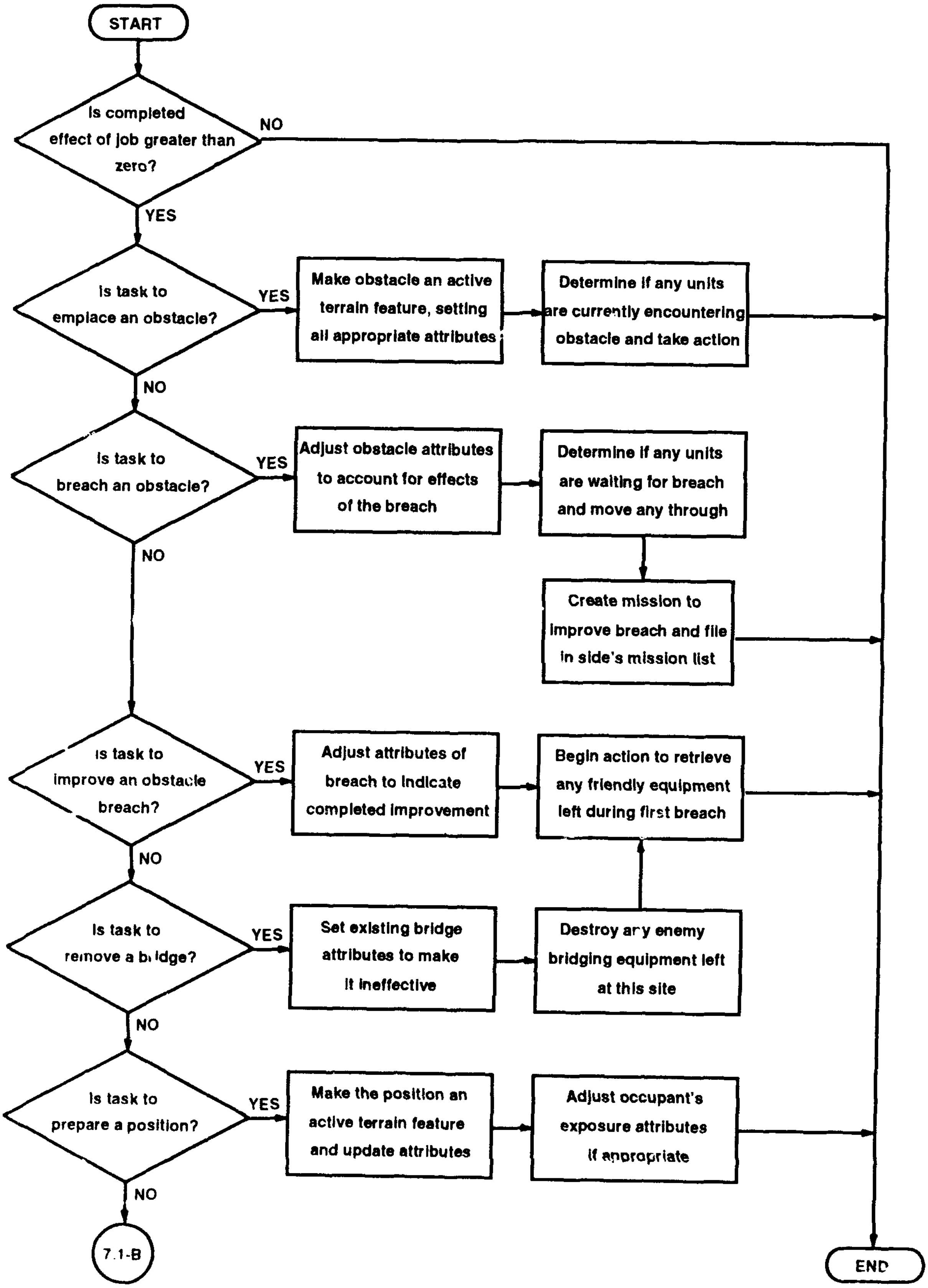
FLOWCHART 6.1 - DISCONTINUE THE MISSION OF A WORK UNIT


START
Is work unit crucial to continuing the mission?
NO
6.1-D ON 6.1
YES
For each active phase from work unit's phase
For each work unit assigned to this phase
Remove work unit from job's phase records
Remove job from work unit's job list and destroy
Is work unit's list of jobs empty?
YES
Does work unit have any assets left?
NO
Destroy all of this unit's data structures and remove it from the set of active units
YES
Determine work unit's path from present location to HQ's location and schedule the move to start
NO
Have all phase work unit's been processed?
NO
YES
Remove phase from job data structures and destroy
Have all remaining phases been processed?
NO
YES
Does current job have any active phases?
YES
Let job's last segment assigned be the last segment of the job's last active phase
6.1-D ON 6.1
NO
Has current job completed at least one segment?
NO
6.1-C
YES
Determine fraction of job complete as end effect of last segment finished + portion; let portion = 0
Register effect of engineer job completion [FLOWCHART 7.1]
Remove corresponding job from staff's list and destroy
Remove job from HQ's set of active jobs and destroy
6.1-D ON 6.1

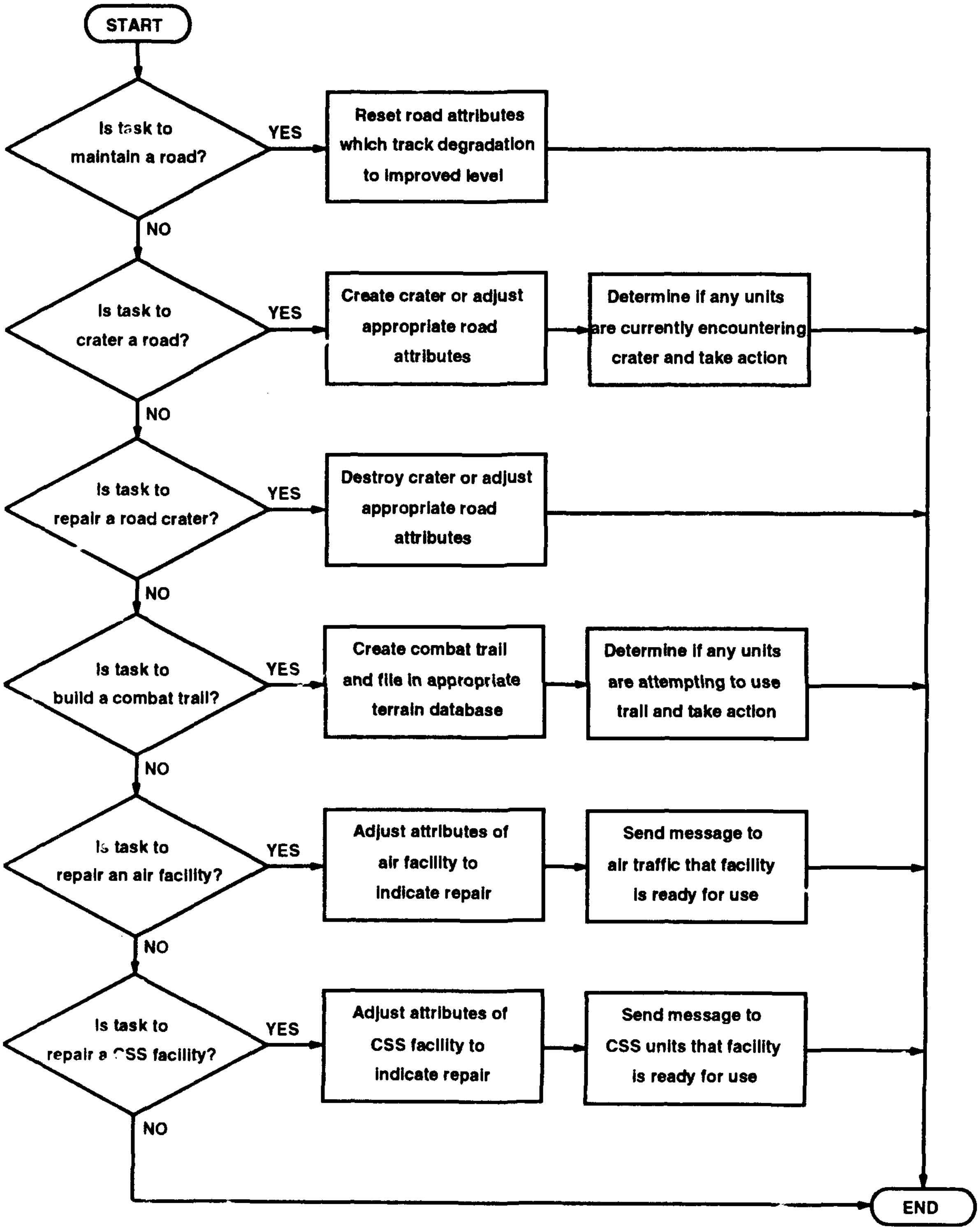
FLOWCHART 6.1-B - DISCONTINUE THE MISSION OF A WORK UNIT (CONTINUED)


START
Can job still be completed in time?
NO
YES
Remove corresponding job from staff's list and destroy
Remove job from HQ's set of active jobs and destroy
6.1-D ON 6.1
Let job's current segment start time be 0
Let job's last segment assigned be 0
File job in HQ's list of pending jobs
Turn off corresponding staff job's active indicator
6.1-D ON 6.1

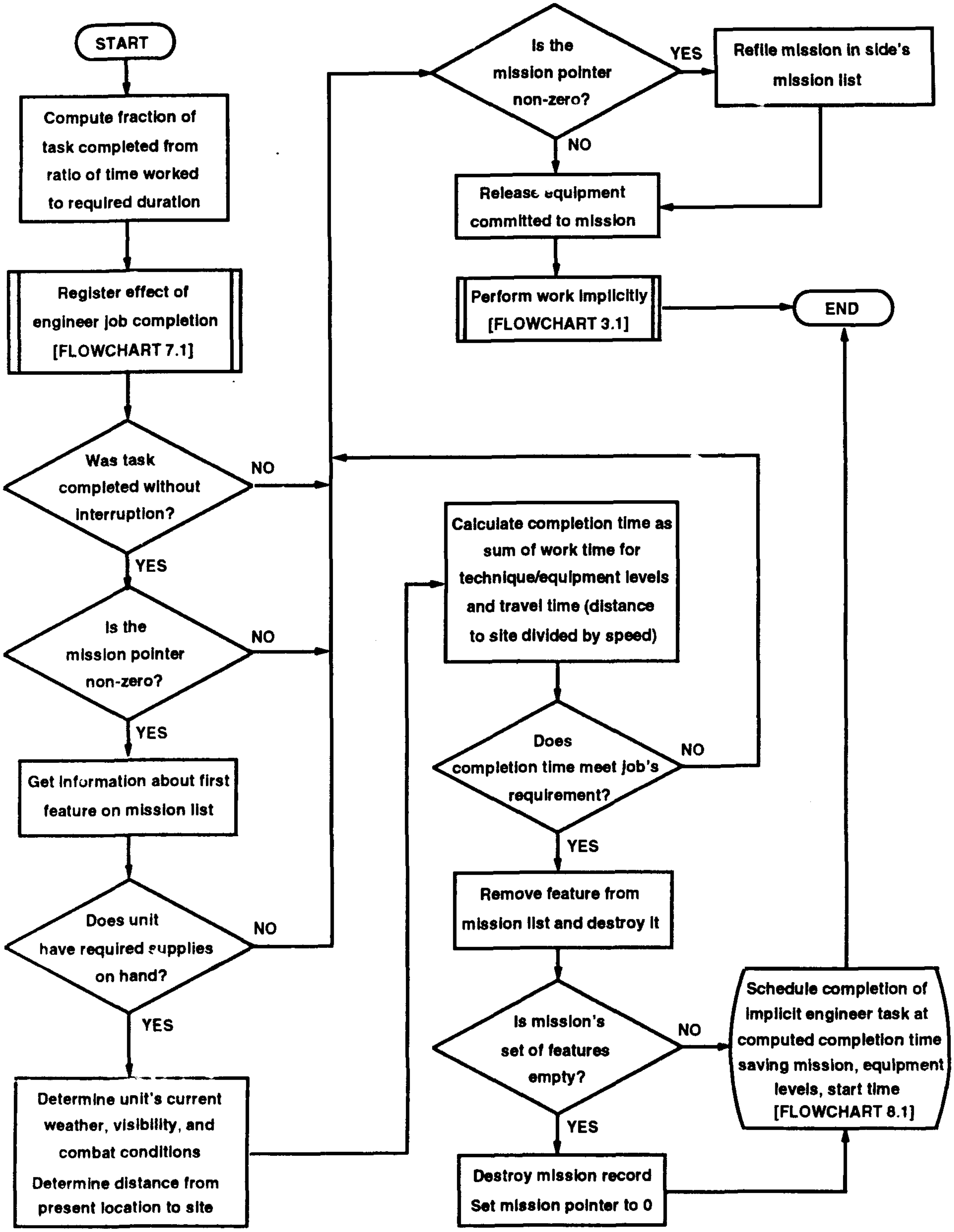
FLOWCHART 6.1-C - DISCONTINUE THE MISSION OF A WORK UNIT (CONTINUED)


START
Is completed effect of job greater than zero?
NO
YES
Is task to emplace an obstacle?
YES
Make obstacle an active terrain feature, setting all appropriate attributes
Determine if any units are currently encountering obstacle and take action
NO
Is task to breach an obstacle?
YES
Adjust obstacle attributes to account for effects of the breach
Determine if any units are waiting for breach and move any through
Create mission to improve breach and file in side's mission list
NO
Is task to improve an obstacle breach?
YES
Adjust attributes of breach to indicate completed improvement
Begin action to retrieve any friendly equipment left during first breach
NO
Is task to remove a bridge?
YES
Set existing bridge attributes to make it ineffective
Destroy any enemy bridging equipment left at this site
NO
Is task to prepare a position?
YES
Make the position an active terrain feature and update attributes
Adjust occupant's exposure attributes if appropriate
NO
7.1-B
END

FLOWCHART 7.1 - REGISTER EFFECT OF ENGINEER JOB COMPLETION


START
Is task to maintain a road?
YES
Reset road attributes which track degradation to improved level
NO
Is task to crater a road?
YES
Create crater or adjust appropriate road attributes
Determine if any units are currently encountering crater and take action
NO
Is task to repair a road crater?
YES
Destroy crater or adjust appropriate road attributes
NO
Is task to build a combat trail?
YES
Create combat trail and file in appropriate terrain database
Determine if any units are attempting to use trail and take action
NO
Is task to repair an air facility?
YES
Adjust attributes of air facility to indicate repair
Send message to air traffic that facility is ready for use
NO
Is task to repair a CSS facility?
YES
Adjust attributes of CSS facility to indicate repair
Send message to CSS units that facility is ready for use
NO
END

FLOWCHART 7.1-B - REGISTER EFFECT OF ENGINEER JOB COMPLETION (CONTINUED)


START
Compute fraction of task completed from ratio of time worked to required duration
Register effect of engineer job completion [FLOWCHART 7.1]
Was task completed without interruption?
NO
YES
Is the mission pointer non-zero?
NO
YES
Get information about first feature on mission list
Does unit have required supplies on hand?
NO
YES
Determine unit's current weather, visibility, and combat conditions
Determine distance from present location to site
Is the mission pointer non-zero?
YES
Refile mission in side's mission list
NO
Release equipment committed to mission
Perform work implicitly [FLOWCHART 3.1]
END
Calculate completion time as sum of work time for technique/equipment levels and travel time (distance to site divided by speed)
Does completion time meet job's requirement?
NO
YES
Remove feature from mission list and destroy it
Is mission's set of features empty?
NO
YES
Schedule completion of implicit engineer task at computed completion time saving mission, equipment levels, start time [FLOWCHART 8.1]
Destroy mission record
Set mission pointer to 0

FLOWCHART 8.1 - REGISTER EFFECT OF IMPLICIT WORK COMPLETION

5 SUMMARY: A LIST OF LESSONS LEARNED

Listed below is a summary of the lessons discussed in this report concerning combat modeling and the representation of combat engineers:

- Maximum flexibility and robustness in model design can be achieved when every behavior that can be controlled in the real world system is mapped to a behavior that can be controlled by the model user in the simulation.

- Massive computer power is not the missing element in the quest to build valid combat models. A poorly designed simulation will not give better results because of improved hardware. (This lesson is an application of two familiar computer maxims: "Garbage in, garbage out," and "Make it work first before you make it work fast.")

- Limiting the model to include only the important variables is essential to the validity of the model, not just its computer run time. The design must capture the essence of each object and interaction.

- Errors in design associated with including inconsequential details tend to occur at the lower levels of the system representation, while errors in design associated with excluding essential details tend to occur at the higher levels.

- Since it is almost a given that any standing combat model will be changed many times, the original structure and all changes to it must be such that the change process maintains rather than undermines the soundness of the design. This comes from a carefully crafted original and from a clear understanding of its structure by those who would offer changes to it.

- The only way to build an easy-to-use full-spectrum combat model applicable to a variety of theaters is to use a high degree of abstraction, with the guiding principle of severely limiting detail and basing the flow of action more on logical concepts than on mathematical calculations.

- The six basic engineer field manuals (FMs 5-100, 5-101, 5-102, 5-103, 5-104, and 5-105) contain most of the fundamental rules required to establish an expert system for combat engineers as they function under Airland Battle doctrine.

- The most crucial time during the life cycle of a combat model for adding engineers is at the very beginning of the design process. If the basic movement and combat algorithms are designed without the engineer function in mind, mistakes will surely be made that will not be correctable after the model is in production use.

- Modeling the effect of engineer task performance is more important than modeling the task performance itself.

- The model cannot measure requirements on engineer effort with any accuracy at all in the absence of a commensurate representation of the effect of the engineer work.

- If the effect of an engineer task cannot be modeled, then the performance of the task should not be modeled. This is because the level and priority of engineer effort are generally functions of the effect and, therefore, cannot be calculated realistically in the absence of an effect.

- The user should be able to model the effects of a functional area without having to simulate the implementing process, i.e., the user should be able to make simplifying assumptions about the way the simulated battle is to evolve.

- High-resolution detail of the processes of the engineer module remains consistent with a model's low-resolution processes as long as the processes themselves do not mix the low- and high-resolution details.

- To decide whether detailed simulation of engineer processes maintains model validity, designers must examine each step where the high-resolution engineer process overlaps the low-resolution portion of the model to verify that the result is acceptable.

- The primary reason for explicitly modeling engineer task performance is to improve the accuracy of estimates of engineer capability, i.e., what engineer tasks can be done and when. The battlefield effect of engineers is in the timing and the effect of their task completion and not in their presence per se or in the simulated performance of their tasks.

- The basic processes in representing engineer task performance are the same for all task types and can be modeled by a single sequence of procedures.

- The representations of the terrain features and facilities that are the focus of engineer work are the keys to how well the model can capture both the capabilities and the contributions of engineers.

DISTRIBUTION LIST

Chief of Engineers:
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CERD-L

CEWES Vicksburg, MS 39180
ATTN: CEWES-GV-A (10)
ATTN: CEWES-EN-A
ATTN: CEWES-GM-L (10)
ATTN: Library

CECRL 03755
ATTN: Library

CEETL 22060
ATTN: CEETL-GL-A

US Army Schools 65473
ATTN: ATSE-CDC-M (20)
ATTN: ATSE-TD (5)

US Army, Ft. Belvoir, VA 22060
ATTN: Engineer Strategic Studies Center Library
ATTN: CEESC-MD
ATTN: Engineer Library

TRADOC Analysis Commanu
ATTN: ATRC-FM 66027 (10)
ATTN: ATRC-WEA 88002

USMISMA 20324
ATTN: ODUSA-OR

USAMSAA 21005
ATTN: AMXSY-CC

NCEL 93043
ATTN: Library

US Military Academy 10996
ATTN: Dept of Military Sci

US Army Command & General Staff College 66027
ATTN: ATZL-SWH

US Army Armor School 40121
ATTN: ATSB-DOTD-CBT

US Army Combined Arms Center 66027
ATTN: ATZL-CAC

Army War College 17013
ATTN: Library

US Government Printing Office 22304
Receiving/Depository (2)

Defense Logistics Studies Information Exchange
ATTN: AMXMC-D

U.S. Army Logistics Management College 23801

Defense Technical Info Center 22304
ATTN: DDA (2)

82
02/92

This publication was reproduced on recycled paper.