[redacted]

MEMORANDUM REPORT BRL-MR-3679

BRL

1938 – Serving the Army for Fifty Years – 1988

AN OVERVIEW AND STATUS REPORT OF MUVES (MODULAR UNIX-BASED VULNERABILITY ESTIMATION SUITE)

PHILLIP J. HANES
KAREN ROSS MURRAY
DOUGLAS A. GWYN
HELEN R. POLAK

JULY 1988

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY

ABERDEEN PROVING GROUND, MARYLAND

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) BRL-MR-3679	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION US Army Ballistic Research Laboratory 6b. OFFICE SYMBOL (If applicable) SLCBR-VL-V	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Aberdeen Proving Ground, MD 21005-5066	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
An Overview and Status Report of MUVES (Modular UNIX-based Vulnerability Estimation Suite)(U)

12. PERSONAL AUTHOR(S)
Hanes, Phillip J. Murray, Karen Ross Gwyn, Douglas A. Polak, Helen R.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP		
12	05		UNIX	structured analysis
			vulnerability assessment	structured design
			physical interaction	configuration management

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

The Modular UNIX-based Vulnerability Estimation Suite (MUVES) is an integrated software system for vulnerability assessment in the UNIX operating system environment. It is being designed to handle a wide variety of current and future targets and threats by use of a generalized physical interaction model. The environment will be sufficiently flexible to support both production and experimental applications and various approximation methods.

Goals of the MUVES system include improved configuration management of vulnerability analysis computer codes, portability of code to a variety of computer systems running the UNIX operating system, and minimization of code redundancy. This software is being developed using a total system life cycle approach to developing and maintaining the software, which emphasizes structured analysis and structured design.

This report is an overview of the current status of the MUVES project. It discusses the development tools being used, presents the results, thus far, of the User Survey and

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Phillip J. Hanes	22b. TELEPHONE (Include Area Code) (301) 278-6651 / 22c. OFFICE SYMBOL

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

Block #19. Abstract (Cont'd)

the Analysis, and discusses the future directions for the project.

Block #18. Subject Terms (Cont'd)

vulnerability analysis
lethality assessment
lethality analysis

TABLE OF CONTENTS

LIST OF FIGURES

I. BACKGROUND AND INTRODUCTION

The Vulnerability/Lethality Division (VLD) of the Ballistic Research Laboratory (BRL) has been facing an increasing number of large-dollar, high-visibility programs. One issue of concern for these programs is the state of current vulnerability/lethality analysis computer codes, their use, maintenance, and extension. It has been noted that many current vulnerability codes, for example VAMP,[1] VAST,[2] and SLAVE,[3] share, at least functionally, many of the same algorithms. Far more code is being maintained than should be necessary, and there may be many versions of essentially the same code. This raises serious questions concerning both efficient use of scarce analyst talent and effective configuration management. In addition, because of the non-structured, monolithic design of these codes it is difficult to extend the current codes with new features.

It was suggested that it would be useful to recode these individual programs into a new environment so that 1] only one module would be written and maintained to accomplish a particular calculation, 2] the new environment would accommodate all existing ground vulnerability codes, 3] it would readily accommodate changes as new algorithms were developed, and 4] it would give the analyst the freedom to change specific modules when necessary, while keeping within a carefully defined and specified analytic framework.

These goals were agreed to by the VLD Branch Chiefs. In early 1985, an Ad Hoc Methodology Working Group was formed with the primary objective of enhancing the state of vulnerability tools. The members of this Working Group were VLD experts in vulnerability analysis. This group met several times in 1985, and each member of the group, as well as other members of the Division, presented individual views on the shortcomings and requirements of vulnerability assessment. Several "white papers" were written discussing a number of key vulnerability issues and requirements for vulnerability modeling.

In the fall of 1985, a team of analysts was assembled to plan and implement a new computing environment for vulnerability analysis based on the findings of the Ad Hoc Working Group. The team made substantial progress in studying the algorithmic layout of current VAMP and VAST codes; however, a major issue delayed work on this project through much of FY86. Some members of the team were directed to develop a high-resolution point-burst code to evaluate stochastic phenomena for comparison with live-fire shots. Although this experience enhanced our appreciation of live-fire effects, it represented a detour from the original plan and produced yet another vulnerability code. Also during FY86, the VLD scrutinized its array of vulnerability codes with respect to their use and maintenance. Configuration management and control of our codes is now a requirement. As attention was focused on these problems, the Ad Hoc Methodology Working Group was reactivated.

---

1. C. L. Nail, E. Jackson, and T. E. Beardon, "Vulnerability Analysis Methodology Program (VAMP) — A Combined Compartment-Kill Vulnerability Model", Computer Sciences Corporation Technical Manual CSC TR-79-5585, October 1979.
2. C. L. Nail, "Vulnerability Analysis for Surface Targets (VAST)- An Internal Point-Burst Vulnerability Assessment Model - Revision I", Computer Sciences Corporation Technical Manual CSC TR-82-5740, August 1982.
3. Douglas A. Ringers and F. Tyler Brown, "SLAVE (Simple Lethality and Vulnerability Estimator) Analyst's Guide," US Army Ballistic Research Laboratory Technical Report ARBRL-TR-02333, June 1981.

A reorganized team began work again on planning the new computing environment. The new team is headed by Dr. Robert Shnidman and consists of the authors of this report. This team prepared a preliminary analysis for a new vulnerability computing environment and presented it to members of the Ad Hoc Methodology Working Group. Called MUVES (for Modular UNIX-based Vulnerability Estimation Suite), it is intended to be an integrated software system for vulnerability assessment in the UNIX operating system environment. It is being designed to accommodate a wide variety of current and future targets and threats through use of a generalized physical interaction model. The design is sufficiently flexible to support both production and experimental applications and various methods of approximating the effects of munitions. This suite is being developed using a structured system life cycle methodology, which will be discussed in Section III.

At this point the MUVES team has completed a User Survey and finished the Analysis phase of the structured system life cycle. The products of these phases include a User Requirements Document[4] and a Structured Specification for the proposed system, which is described in Section IV. Following these phases, the team will continue with the Design, Implementation, and Testing phases of the project.

In this report, we present the status of the MUVES project. In Section II we will discuss the overall goals of the MUVES project. Section III gives an overview of the structured system life cycle methodology being used to develop this system. Section IV will discuss the details of the data flows and process descriptions for MUVES. Section V lists the new requirements for vulnerability analysis that are being addressed by MUVES. Section VI presents the advantages of MUVES over previous methodology. Possible future directions of the MUVES project are discussed in Section VII.

---

4. Aivars Ozolins and Paul Tanenbaum, "Assessment of Vulnerability Analysis Capabilities and Requirements," 28 Feb 86.

II. GOALS

The major goals of the MUVES project are the following:

1. PROVIDE CURRENT CAPABILITIES IN THE UNIX ENVIRONMENT

The new suite of codes will provide many improvements in our vulnerability analysis capabilities; however, it must also provide all capabilities available in the existing ground vulnerability codes.

There are now many classes of computers at the BRL which run some variant of the UNIX operating system: single-user graphics workstations, a variety of minicomputers, and a Cray X-MP supercomputer. This trend is expected to continue for computer acquisitions in the foreseeable future. Therefore it is important that the new suite of vulnerability codes work within the UNIX environment. This will allow a user to work on the computer that is appropriate for a particular task, with the capability of switching easily to a different class of computers for a different application, since the suite can readily be made available on all these systems.

2. USE MGED AND RT FOR TARGET GEOMETRY

The MUVES environment will use the Multi-device Graphics EDitor (MGED) for solid modeling and the Ray Tracer (RT) for interrogating the geometry. The MGED program and RT library already exist on BRL UNIX systems; by utilizing these facilities the MUVES project can avoid a substantial amount of development work that would otherwise be necessary.

Currently, several vehicle target descriptions have been done using MGED. A centralized library of all available target descriptions is being formed. Additionally, older target descriptions in COM-GEOM format can be converted to MGED-compatible target descriptions with existing utility software.

3. CONTROL PRODUCTION CODE DESIGN AND MAINTENANCE

As mentioned earlier, the Vulnerability/Lethality Division has set as a goal the improvement of configuration management of our vulnerability codes. Currently, there are often so many customized versions of a particular code that it is possible for different analysts to run the "same" code on a particular threat/target combination and get quite different results.

MUVES, while providing hooks for users to run experimental applications, will be more tightly controlled with regard to modifications. There will be an "official" supported version of MUVES maintained in a central directory (identical across all computer systems). This will be used for all routine production runs in order to maintain consistent results throughout the Division. Vulnerability analysts will have the capability of adding experimental interaction modules as separate processes for their own use. These modules may eventually be integrated into the supported MUVES system after being reviewed for compliance with software maintenance guidelines.

The existing Source Code Control System (SCCS), a collection of programs that runs under the UNIX operating system, will be used to control and account for changes to MUVES software and documentation. Additional record-keeping procedures will be implemented to provide required configuration control information.

4. PROVIDE USER-FRIENDLY INTERFACE

There will be an easy-to-use interface between the analyst and the MUVES computational software. The interface will help to automate inputs to the system and provide record-keeping functions. The structured life cycle approach is being taken to develop this system; thus, documentation for the system will be written before coding is begun. Full and clear documentation will be provided as well as any necessary training.

5. PROVIDE FLEXIBLE USER-SPECIFIED PROGRAM INTERACTION

Computer graphics technology has been advancing at an amazing rate in recent years, and our computer-aided geometric techniques have been improving; however, we are still far behind current state-of-the-art in the input and output routines being used with our vulnerability codes. We hope to significantly improve that situation with MUVES.

The analyst working with MUVES will be able to work with a hierarchy of menus for such things as selecting input parameters, specifying analysis methods to be used, and specifying desired outputs. Graphic displays and menu-driven inputs will ultimately be developed for interacting with MUVES, in addition to traditional methods of keyboard input entry. There will be default values for most inputs that can be used for a variety of common situations. The user will be able to edit the input values and save them for use in a later run, so that it will not be necessary to go through the entire input process each time.

Care will be taken to ensure that MUVES facilities can be accessed from a wide variety of present and future computer terminals and interactive workstations. Specific interfaces to exploit capabilities of special high-performance devices will be developed after the more general initial interface.

6. MINIMIZE NEED FOR ANALYST PROGRAMMING

Within MUVES, unique requirements for vulnerability analyses should usually be handled without having to modify software. The code will be general enough so that in most situations an analyst would only need to modify input parameters or request different options of the user interface to obtain the required results. (Unusual situations may require extension of MUVES, which is being designed to facilitate such adaptations; see paragraph 8.)

In addition, due to the structured, modularized nature of the MUVES code, the difficulties of code maintenance will be minimized. When changes are required, one will need only to change the individual module or modules affected, rather than looking through the entire code for embedded interacting sections that need to be changed. By making modifications only to modules with well-defined interfaces, one avoids undesirable side effects that may occur when a change in one section of the code has an unanticipated impact elsewhere in the code.

7. MINIMIZE CODE REDUNDANCY

In many cases the same penetration algorithm has been implemented differently in two or more versions of the same code. Thus, a change to the basic algorithm means that each instance of the algorithm in each version of the code must be found and updated (most likely in slightly different ways). A goal of MUVES is to have only one module written and maintained to accomplish a particular calculation, thus avoiding the problem of different implementations of the same algorithm.

8. FACILITATE EXTENSION OF PRODUCTION CODE

In all phases of the analysis for MUVES we have been sure to make provisions for future extensions of the code. Thus, as new algorithms are developed, incorporating them into MUVES will be straightforward. The MUVES system will provide an environment in which a vulnerability analyst can explore new situations, new methods of analysis, and experimental applications without requiring a separate vulnerability code.

III. METHODOLOGY

The techniques being used to develop the MUVES analytical environment are the result of recent advances in computer science which emphasize the total system life cycle and use structured approaches to all phases of system development.[5] The primary reason for using this methodology is to minimize the total cost of developing and maintaining software. This is accomplished by:

- Investigating the requirements for a proposed system before investing time and money to develop the system,
- Partitioning the overall procedure into manageable tasks with clearly defined interfaces,
- Structuring the design of the code so that it is easy to understand and, therefore, easy to maintain and modify,
- Thoroughly documenting these efforts as they are performed, so that written code will be directly related to its documentation and to its desired function,
- Preparing skeleton versions of code early in the life cycle for quality assurance, and
- Implementing lower-level modules in parallel.

The structured system life cycle methodology begins with a User Survey to determine whether or not a new system is needed. It does not end until the debugged and tested software is retired from service. The techniques that are of primary interest in this report are those yielding full documentation for the proposed system. For completeness, we will briefly describe all phases of the structured system life cycle methodology. These phases are User Survey, Structured Analysis, Structured Design, Top-Down Implementation, and Maintenance.

1. USER SURVEY

In the User Survey, we determine the requirements for a new system based on the current procedures and known deficiencies of the current system. Everyone using the current system is given a chance to offer comments, suggestions, and criticism. The user survey is an iterative process which results in a broad range of requirements and desired features which are used to guide the Structured Analysis phase.

2. STRUCTURED ANALYSIS

In the Structured Analysis phase we examine the results of the User Survey and formulate a Structured Specification for the new system. The Structured Specification is a graphic, concise, top-down partitioned model of the functions essential to the envisioned system. By "top-down

5. Only a brief overview of these techniques can be given here. For more information on these concepts, see E. Yourdon, "Managing the System Life Cycle", Yourdon Press, 1982; T. DeMarco, "Structured Analysis and System Specification", Yourdon Press, 1978; and M. Page-Jones, "The Practical Guide to Structured Systems Design", Yourdon Press, 1980.

partitioned" we mean that the overall operation of the system is divided into several tasks, each of which may then be subdivided into smaller tasks. This partitioning proceeds until the entire system is broken down into small, manageable processes which a programmer can deal with easily. This partitioning is accomplished by examining the desired output data and the available input data, then determining the transformations required to produce the former from the latter. Those transformations are the tasks which we partition into bite-size chunks for the programmers. As this top-down refinement proceeds, all terms used in the descriptions of the data and transformations are formally defined, and algorithmic descriptions of the elementary processes are produced.

The tools used to document these data flows and transformations are data flow diagrams, the data dictionary, and transformation specifications. Together, these form the Structured Specification.

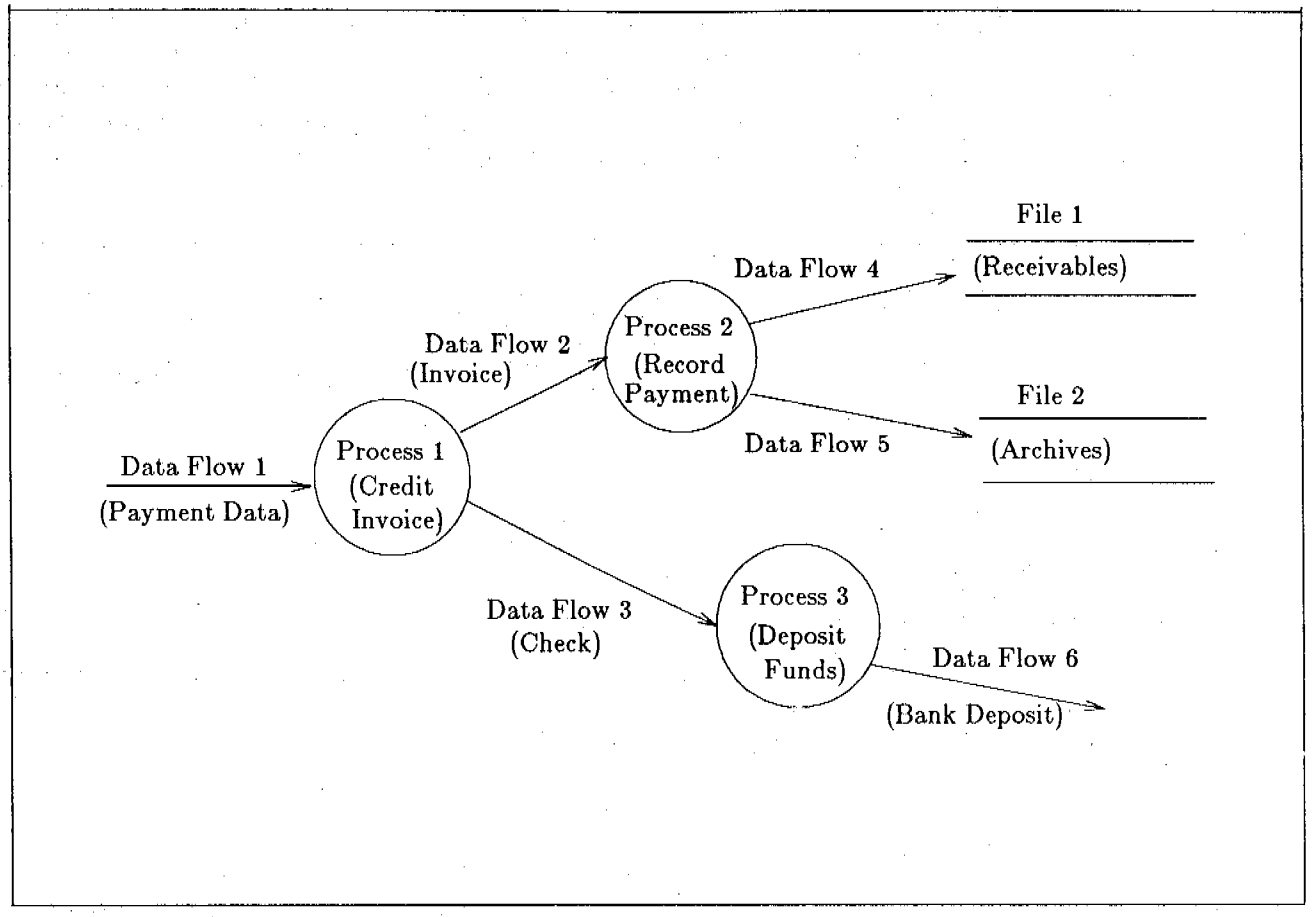

Figure 1. Example of a Data Flow Diagram.

On a data flow diagram (figure 1), each flow of information is depicted by an arrow-tipped line segment showing the direction of flow, each transformation is represented by a bubble called a "process", and data files are represented on the data flow diagrams by two parallel horizontal

lines. One might think of the data flow diagram as a pipeline of information with a steady stream of data flowing through it; the processes are joints in the pipeline, and files are eddies in the current. No control flow is shown on a data flow diagram, because it adds complexity to the formulation of a system and is not needed to define the flow and transformation of the data. At this stage of analysis, the goal is to define what the system will do with the data, but not how it will do it.

The top-level data flow diagram shows the major processes and the data flowing between them and into or out of the system. Each process on the top level diagram itself consists of several processes which, in turn, may be broken into further processes. A data flow diagram is drawn for each such partitioned process, showing the processes and data flows which compose that larger process. The resulting set of hierarchical data flow diagrams allows the software developer and the user to discuss the system at various levels of detail without having excessive information in any one diagram.

To help specify interfaces between processes and to avoid using ambiguous terms on the data flow diagrams, we define in the data dictionary the contents of each data element on the data flow diagrams. Definitions in the data dictionary (figure 2) are precise explanations of each type of data and the combinations of those data. Complex data flows are defined in terms of other data elements, which are defined in detail elsewhere in the dictionary. This permits exact definitions of all necessary data types in a concise and readable fashion.

PAYMENT-DATA	=	Customer-Name + Customer-Address + { INVOICE-NUMBER + Amount-of-Payment }
INVOICE-NUMBER	=	State-Code + Customer-Account-Number + SALESMAN-ID
SALESMAN-ID	=	Five digit number

Figure 2. Example of Entries in a Data Dictionary.

We have used the convention that capitalized words in the data dictionary definitions are terms that are further defined in the data dictionary. The notations used in the data dictionary are defined as follows:

= means IS EQUIVALENT TO.

+ means AND.

[ ] means EITHER - OR; i.e., select one of the options enclosed in the brackets.

{ } means ITERATIONS OF the component enclosed.

( ) means that the enclosed component is OPTIONAL.

The third section of the Structured Specification contains transformation descriptions, where we describe the policies used to perform the lowest level data transformations. The interior functions of each of these processes are detailed in a one-page description called a mini-specification (figure 3). Mini-specifications are written in a concise subset of the English language called Structured English.

Process 1: Credit Invoice

```
If the amount of the invoice exceeds $500,
    If the account has any invoice more than 60 days overdue,
        hold the confirmation pending resolution of the debt.
    Else (account is in good standing),
        issue confirmation and invoice.
Else (invoice $500 or less),
    If the account has any invoice more than 60 days overdue,
        issue confirmation, invoice and write message on the
        credit action report.
    Else (account is in good standing),
        issue confirmation and invoice.
```

Figure 3. Example of a Mini-Specification.

The Structured Specification is the basis for the design of the system. The Structured Analysis phase of system development can also yield estimation heuristics by which to judge the progress of the project and aids for generating acceptance tests.

3. STRUCTURED DESIGN

Structured Design is a strategy for converting a Structured Specification into an easily implemented, top-down design. Modules are arranged into a hierarchy in such a way that they have a one-to-one correspondence with the modules in the eventual system of programs. This hierarchy is represented graphically in a Structure Chart, which shows the modules and the

relationships among them. The Structure Chart contains most of the control information required to implement the actual system. As the goal of the Structured Specification is to define what the system is to accomplish, the goal of the Structure Chart is to define how the requirements will be met.

Two important objectives in Structured Design are to minimize coupling between modules and to maximize internal module cohesion.

Coupling is the extent of interaction between any two modules. Strongly coupled modules are usually not separately maintainable; it is difficult to modify one without requiring modifications to the others. Large numbers of coupled modules in diverse parts of a program can become a programmer's nightmare as he tries to anticipate all of the ramifications of correcting even a single line of code. This can be equally frustrating for the user who stumbles across those sections the programmer failed to correct. Such difficulties are well-known to managers of hastily planned and implemented systems.

Cohesion is the degree to which the tasks performed in one module are related. Modules have low cohesion when a single module performs several unrelated tasks, bound together by weak dependencies. Such modules are tedious to modify because their structure makes it difficult to identify the portions which actually need to be rewritten to effect the desired changes. For instance, an initialization module performs a wide variety of operations which often have no real relation except that they are done near the beginning of the program. Such cohesion is called temporal cohesion and is generally undesirable. Explaining the impact of all of these operations on the rest of the program would be as much work as putting them in the appropriate places initially.

A "good" type of cohesion is functional cohesion. In a module with functional cohesion, every processing element in the module is essential to the performance of a single function. If a module is functionally cohesive you should be able to describe what the module does in one simple, coherent English sentence.

There is usually a strong relationship between coupling and cohesion. A system whose modules have low cohesion will probably have strong coupling, and vice versa.

During the design phase, we also consider the packaging of the system, wherein the hardware constraints are taken into account. In addition, we try to include as many items from the user interface "wishlist" as possible without adversely affecting performance of the essential functions.

The Structured Specification and the Structure Chart compose the full documentation for the system, including details of program hierarchy, module contents, module interfaces, file structures, and the user interface.

4. TOP-DOWN IMPLEMENTATION

Top-Down Implementation begins with coding and interfacing for the top level modules, and proceeds to the addition of middle and lower level modules. Early in this phase, a walk-through may be held with the users to determine whether the system will, in fact, meet user requirements. The modules are written with strict adherence to the mini-specifications, while their interfaces and control flow are determined by the Structure Chart. Thus, we insure that the code does exactly what the documentation says it should do. The principles of structured

programming are easy to follow, given the logical structure and thorough documentation of the system.

5. MAINTENANCE

Inevitably, it will become necessary to modify any system (no matter how well designed) to meet the changing requirements of the users. When this happens, the proposed changes are to be analyzed as carefully as the original system and the results of this analysis then incorporated into a revised Structured Specification. The system is then re-designed and displayed in a new Structure Chart. The code is then modified in accordance with the revised Structure Chart and is consequently already as fully documented as the original version. The structured nature of the documentation and the partitioned processes makes it easy to pinpoint those parts of the system that will be affected by a proposed change. This reduces the time and effort required to implement modifications. The relative ease of maintaining such a system even after the original authors have gone on to other projects is one of the primary advantages of the Structured System Life Cycle Methodology.

This approach to software development is designed to avoid many problems inherent in conventionally managed software projects. In the years since the introduction of these structured techniques, its practitioners have observed significant reductions in common software problems such as schedule overruns, expensive maintenance, and systems that are of little or no use to the customers. Early in the project, these techniques can help managers determine the tasks that need to be done and evaluate more accurately how long they will take. They encourage consistent analysis, design, and coding practices which reduce the cost of maintaining the code after it is written. And, if the User Survey and Analysis are done properly, they insure that the eventual product does what the users actually need.

IV. PROJECT DESCRIPTION

In this section of the report we examine the data flow diagrams, the primary means of defining the data flows and transformations for MUVES, a generalized software system for performing vulnerability analysis. In developing these data flow requirements, we followed a consistent overall philosophy toward the problem of assessing the remaining utility of a target damaged by some sort of threat. This approach has helped us clarify many of the conceptual difficulties which have plagued previous vulnerability analysis efforts. The first aspect of this philosophy was to generalize the statement of the problem in such a way that most vulnerability problems could fit into a single, generic framework. That framework is stated in the following sentence: "An object, called the threat, is thrown at another object, called the target, in such a way that its trajectory can be approximated by a ray; some sort of damage occurs to one or both objects as a result of the interaction between the two objects, and possible new threats are created; then the utility of the damaged target is assessed".

Among our major concerns, therefore, are characterization of threats and targets and the events occurring at the moment of impact of a threat with a target. In order to make the interaction aspect of the problem more transparent to the analysts and easier to change as new data becomes available, we have isolated consideration of all threat-target physical interactions into one process within our framework. We have named this process the "interaction module"; however, it is important to realize that this is not, in fact, one module, but a large number of modules each of which performs the calculations for a specific type of threat interacting with a specific type of component within the target. In addition, the components of interest and the manner of evaluating damage to them varies in different types of analyses or approximation methods. Therefore, there will be a different family of these modules called upon to do the calculations in each particular analysis, depending on the approximation method chosen by the analyst. Many of these modules are likely to belong to more than one such family, especially frequently used ones such as those for armor perforation calculations. In this manner, we hope to eliminate redundancies evident in earlier vulnerability programs while providing a previously unavailable flexibility.

This philosophy places certain restrictions on what can be done by this system. Unfortunately, MUVES will not be capable of doing everything that any vulnerability analyst in the world might wish. For instance, no projectile with a curving trajectory can be represented under this scheme, since all motion is tracked by ray-tracing, and rays are inherently straight. Also, such events as diffusion of gases and temperature radiation are not feasible, since they consist of fronts of molecules or energy moving through varying media. Similar difficulties arise with blast and shock effects. These may, however, be approximated using large numbers of rays radiating from a single point and moving through only one medium, air. This technique is computationally intensive, but could be used to study these phenomena for complicated target geometries until better methods can be devised.

Since the Structured Analysis phase of a project is intended to completely and accurately define the needs of the users of the eventual product, we have attempted to produce a generalized description covering everything that must happen in order to analyze the vulnerability of any desired target versus any direct, impacting threat. Previous efforts to revise vulnerability methodology have frequently missed this generality by concerning themselves with hardware limitations or current techniques at too early a stage in the analysis. We have described a system which should meet all current analytical requirements with few changes in the necessary inputs, but which eliminates most of the redundancies and streamlines the entire procedure. That system is described by the data flow diagrams in Appendix A. The process definitions and

data flow definitions associated with those diagrams are provided in Appendices B and C, respectively.[6]

The first figure in Appendix A is *not* a data flow diagram. It is, rather, a tree structure chart intended as a visual aid for those who are unaccustomed to reading data flow diagrams and who wish to study the overall hierarchy of the system. As such, it is not exhaustive; several of the lower-level processes were left out to avoid cluttering the chart. Also, it gives no information about the data used within the system or how those data pass between processes.

Upon examination of the first data flow diagram, it should be apparent that MUVES is partitioned into three major processes, each of which serves a single, generalized function. They are not necessarily intended to be separate "processes" in the traditional UNIX sense of the term, although they may be if that is the decision made at the implementation stage. Instead, they are separate conceptual entities each of which transforms data in a particular fashion, then passes the transformed data to the succeeding processes.

The first major process in MUVES is called Generate Initial Threats; it is the "throwing objects" aspect of our general framework. This process accepts a specification from the analyst concerning the specific threats, plus the pattern and density of shots to be fired at the target. It then calculates the start points and directions for the collection of shots to be fired at the target and an averaging weight for each of them based on the user-specified dispersion. Finally, it looks up information about the desired threats and attaches it to the calculated trajectories.

The second major process in MUVES is called Simulate Physical Interaction; it studies the "interaction between the two objects." This process uses the Ray Tracer to interrogate the target geometry, finding the intersections of the threat trajectories with the target components and the surface geometry at each intersection. It then calculates the results of the impact of the threat with each individual target component located by ray tracing. Generally, this comprises damage to the components plus degradation and/or deflection of the threat, as well as some possible secondary threats (for example, spall) resulting from that impact. These calculations are usually empirical predictions based upon analyses of a large quantity of data gained from experiments performed at BRL and other locations during the last couple of decades. This process only calculates the physical damage done to each component in the target; it makes no estimation of the significance of this damage with respect to the target's ability to perform its mission.

The third and final major process in MUVES is called Assess Remaining Utility; this is where "the damaged target is assessed." This process evaluates the functionality of the damaged target after all interactions have taken place. Initially, MUVES will be tailored to address ballistic perforation, although the basic framework can be extended to address several other conceptually different types of threat. The damage to each component at one or more times after the interaction of the main threat and the target, is condensed by the use of the Component Damage Function into a numeric estimation of the loss of that component. This can be expressed as the probability of total loss or as a fraction of the component's full capability.[7] These component

6. These initial specifications are subject to revision as the design effort proceeds.

7. Although often treated in the same fashion or even used interchangeably, these concepts are different and generally do not produce equivalent results. Discussion of this issue is beyond the scope of this paper, but MUVES can be used to accommodate either approach and, perhaps, may help clear up the differences.

damage values are combined into system degradations, which are then combined into "kill values" representing the entire target's loss of utility with regard to specific missions. These combinations are calculated according to information contained in deactivation diagrams and damage assessment tables, which are collections of dependencies and weights compiled from the judgments of various engineers and tank commanders.

The diagrams in Appendix A give a representation of the data flow requirements for the system and the processes operating on those data flows, but they are still incomplete. They do not give any indication of the exact composition of the data, nor of the internal logic of those processes. Our complete definitions of the data flows are listed in Appendix C. We have also written short, general definitions for each of the processes to help us understand their specific functions; these definitions are given in Appendix B. However, we will have to write more formal "Mini-Specifications" for each process in order to fully define its function in the context of the MUVES system. When these are finished, they will complete our study of the data flows for a generic system for performing vulnerability analysis in the UNIX operating system environment.

Another very important aspect of the project which we have not yet formally addressed is the user interface, that is, the set of queries and menus with which an analyst communicates with the system. Before doing that, we must fully define the processes and data flows required to do the necessary specific calculations. When we have accomplished that, we will begin work on the data structures and file formats needed for communication with the external environment. Then, knowing what is needed and what it should look like, we will begin work on the user interface. Although we have not defined any of the details of the user interface, we do have some guidelines and goals. These are described in section V as part of the requirements for the MUVES system.

V. REQUIREMENTS

The following are the vulnerability issues and new requirements addressed by the MUVES design. They were derived from notes of the meetings of the BRL/VLD Ad Hoc Methodology Working Group and from the "Assessment of Vulnerability Analysis Capabilities and Requirements" by Aivars Ozolins and Paul Tanenbaum:

1. USER INTERFACE

One of the primary requirements for the user interface is to keep track of what the analyst does during any particular work session. This will allow any analyst to refer back to previous sessions to compare results or to edit old input files in order to re-use them on runs which vary only slightly from the previous conditions. Also, there will be an electronic notebook to record any comments, such as the reasons for any variations from the "standard" way of doing things, or simply to keep track of what has been done on a particular analysis. Only information specific to the current analysis will be saved in this notebook; a simple "keystroke log" is neither desirable nor necessary. This notebook will be cross-referenced with the saved input files and results.

We will be publishing standard formats and data structures for the interface which should aid detection of errors in the inputs for specific runs and in the target and threat characterizations. This interface will also allow the analyst to alter the program inputs "on the fly." In other words, he will be able to prepare and change them in any desired order, rather than having to follow the program's lead or restart the entire session whenever a mistake is made. We will accomplish much of this through use of a hierarchical menu system, which may eventually have a mouse interface for those terminals and workstations that support them. There will be default values for a large number of common situations, which each analyst will be able to edit in order to tailor MUVES for his own needs and preferences. This will allow the analyst to enter only those parameters which he wishes to vary on a particular analysis, rather than being led through a time-consuming series of queries each time he runs the program.

We have also, after numerous discussions with experienced vulnerability analysts, determined that nearly every analyst must at one time or another have access to intermediate results in order to assure meaningful final results. Therefore, we have identified several key flows of data which contain the information of greatest use to the analyst for detecting errors or understanding trends in the interactions of threat and target. The analyst will be able to execute selected portions of the system in order to examine these results before continuing the calculations. We will provide various graphic tools to display these results in a meaningful fashion; in addition, all currently used output formats will be available to ensure continuity with past experience. If it should become necessary due to physical phenomena not considered in the available modules, the analyst will be able to edit the intermediate results as appropriate and then run the remainder of the analysis with this altered data. (Of course, such actions will be recorded in the "electronic notebook".)

2. GEOMETRY

In order for this system to operate properly, it will be necessary to institute standards for target characterizations. The programs will require that certain information be in certain places, and if it is not, the analyst will be told of the deficiencies so that they may be corrected. This should help to stabilize our library of target descriptions and related information, correcting a

problem which has been with us for a long time.

We have also provided the means to circumvent many of the problems associated with modern armor, especially when attacking it at edges and corners. The use of altered geometry and some specialized interaction modules should allow the analyst to deal with most of the technological advances in modern armor. In those circumstances where this is not sufficient, or when there is no empirical model in existence, the interaction modules may be designed to recognize encounters with the difficult armor and to activate a portion of the user interface which will allow the analyst to deal manually with those cases.

3. ON-DEMAND GEOMETRY INTERROGATION

One significant advantage that MUVES has over earlier vulnerability assessment methodologies is the availability of a dynamically callable ray-tracing utility. This allows it to perform analyses that were previously impractical. For instance, the analyst can, if he wishes, follow the path of a deflected main penetrator. The ray-tracer can do this quite easily, since it merely needs to know the angle of deflection (which can be calculated in the appropriate interaction module) and the last exit point to use as a new starting point. The effects of ricochet can be examined in the same fashion. These well-known effects were not simulated previously due to the constraint of determining all of the geometrical information as a batch process, without making intermediate calculations.

Some other phenomena that have been avoided in the past because of the constraint of calculating all rays before determining the effects of the impacts are multiple burst points and non-standard fuzing. Multiple burst points were simply not worth the extra computation required to calculate the spall cones in advance when relatively few would actually be used. MUVES avoids this by generating spall cones only where perforations are predicted. Past attempts to add non-standard fuzing such as proximity fuzing and fly-over shoot-down into the old methodology have resulted in a great deal of frustration for those attempting to make the necessary changes. Our methodology will be able to perform these calculations efficiently, possibly by treating the sensor as an initial "pseudo-penetrator" which triggers the munition as a "generated threat" when it detects the target.

4. PHYSICS

The combined features of the MUVES system will allow analysts to approximate physical phenomena with an accuracy unattainable with currently existing methods. The modularity of this approach to vulnerability analysis allows the calculations for any particular type of interaction to be modified without impacting the rest of the system. Thus, any advances in modeling perforation mechanics or other effects can be incorporated and tested as they become available, avoiding long delays for entire programs to be modified to accommodate new methods, and, perhaps even more importantly, avoiding an unmaintainable proliferation of slightly modified codes. Unusual effects such as crossing velocity can be investigated using dynamic ray-tracing combined with new interaction modules designed to accept skewed penetrators. Recent innovations in armor technology can be approximated with altered target geometry and new interaction modules specifically designed to deal with these technological advances.

There is an on-going effort to improve our understanding of behind-armor effects following perforation of armor by various munitions. As new behind-armor debris models are developed, they can be included in existing armor perforation models by making changes only to the

interaction modules for the munition in question. These changes could be made by the scientist or engineer performing the actual data analysis without impacting the integrity of MUVES for production use. After thorough testing to verify that they accurately reflect observed results, the changes can be incorporated as additions to the "official" production version of the code.

There have been some recent attempts at the BRL to estimate the effects on a vehicle of blast from an explosive charge or from perforation of the armor on that vehicle. A blast wave could be emulated using a dense collection of rays radiating from a point to approximate the motion of the pressure wave front through air. Using this technique, the effects of a pressure wave can be calculated at several "detector" locations in order to determine blast effects over the entire vehicle.

There is now some work being done to determine the effects of damage on the performance of various components in certain vehicles. These tests may drastically alter the existing component damage criteria, which were developed from analytical models of the components. These new criteria may not be completely compatible with the old methods of analysis, yet in the past they would have been forced into existing formats in preference to changing the code. MUVES, on the other hand, uses a generalized interpretation of damage results which provides the opportunity to change the methods used for the estimation of component damage as new live-fire data become available.

5. SYSTEMS ANALYSIS

Many of the difficulties involved in combining component damage into overall vehicle damage have also been addressed by the MUVES system. The first improvement is quite simple; we merely altered the concept of the damage assessment table to that of a damage assessment list. The list in the new methodology corresponds to a column in the old tables, but it allows the freedom to add more lists for differing environments or new kill criteria without troublesome format alterations.

Another feature of MUVES is the ability to start and stop the flow of data at any reasonable point in the calculations. This provides several capabilities that were simply impossible in the past. For instance, multiple hits can be estimated by combining lists of damaged components for separate shots into one master list for the multiple shots in question. One can also vary the directions of the entry vector and resultant spall rays for a single shot location to obtain a statistical sample for comparison with the stochasm of live-fire shots as in the SQUASH model. Statistical samples can also be used to provide confidence intervals for the shot results. The capability still exists within MUVES to use the deterministic methods of VAST if desired, simply by stopping the calculations immediately after ray-tracing and saving that information in a file for further calculation at a later date. Also, any of these procedures can be specified at the beginning of the session without any changes to the code.

One other advantage is that all of the interactions use the same target description with different selected groups at a particular time. This allows the analyst to compare the results of different approximation techniques (for example, point burst vs. compartment model) on exactly the same set of shotlines. It also allows some new features such as the development of new compartment correlation curves from a detailed, stochastic point burst evaluation of the target.

VI. BENEFITS

The following are the significant advantages we perceive of MUVES over the previous methodology.

1. USER-DEFINED SHOT PATTERNS

One of the most readily apparent improvements of MUVES over the current methodology is that the analyst will be able to choose from a variety of available shot patterns, rather than being constrained to a single, pre-determined pattern (such as a simple grid). Initially, there are plans for three different shot patterns: a rectangular grid with flexible parameters (including perspective rays), randomly distributed patterns (such as a bivariate Gaussian) around a single aim point, and spherically distributed start points. This allows the analyst to choose a shot pattern appropriate to the problem, rather than trying to fit his problem into the available solution. Some of the tools to support this approach already exist, while others are under development; however, the underlying facilities will be general enough to allow anyone to develop his own special-purpose pattern and use it for the analysis. Additionally, environmental view averaging, such as cardioid weighting, will be supported. Such weighting can be applied to collections of any of the shot patterns described above.

2. INTUITIVE REPRESENTATION OF PHYSICAL INTERACTIONS

Data flow diagram 2.2 shows the approach taken to modeling physical interactions in the MUVES environment. Rays (shotlines) intersect components which incur damage at those intersections. Our concept of a component is somewhat different from that in current use, however. In MUVES, a component is any object which can be analyzed as a single unit with respect to its behavior on interaction with a threat. Thus, a compartment from a VAMP-style analysis will be considered as one (rather large) component. Smaller components, those which currently exist in target descriptions and are used in point-burst analyses, still fall under that category, so little has changed in how they are treated. This gives the target describer the flexibility to build large components for a coarse analysis out of several smaller components intended for more detailed analysis methods. This then permits the analyst to generalize certain aspects of the analysis while retaining detail only where it is necessary and to use a single target description for any method of analysis, changing only the grouping of the objects to achieve the desired result. There is a limit to how far this approach can be pushed, however. Some analysis methods may require geometric detail that would unduly slow computations for other approximation methods. MGED offers some support for maintaining a collection of geometric descriptions with varying degrees of resolution for a single target.

3. ON DEMAND ANALYSIS OF SUBSIDIARY THREATS

There is a feedback loop in this system which returns all generated threats (for example, spall or deflected main penetrator) to the Ray Tracer when they are encountered in the Interaction Module. This technique has a number of advantages over the current practice of calculating the trajectories of potential spall rays before any analysis occurs. First, a large amount of computation is saved by not tracing spall rays when the main penetrator fails to defeat the armor. Second, this allows the analyst to add unusual phenomena which have heretofore been ignored, such as deflected residual penetrators, ricochet, and second-order spall (this of course presumes that algorithms will be available to calculate the necessary parameters for representing

these phenomena).

4. ISOLATION OF THREAT-COMPONENT INTERACTIONS

Each type of threat-component interaction will be performed by a single module within the final MUVES implementation. Thus, each such module will have a single, well-defined function which it performs upon demand. Therefore, MUVES need only call the family of modules appropriate for a particular analysis. This also has the benefit of making the modules easier to understand individually or as a complete package. For this approach to work properly, the interfaces to these modules *must* be clearly and cleanly defined, in order to make it possible for the controlling processes to "plug in" the appropriate modules as they are needed. The standardized interface will allow the analyst to insert special components, experimental materials, and new empirical models into the system without revisions to the rest of the code. This prevents unwanted side-effects in other portions of the system, since information transfer occurs only at certain points and in a well-defined manner. This is one way that MUVES gains maintainability over the plethora of currently existing vulnerability analysis computer codes.

5. ISOLATION OF ENGINEERING APPROXIMATION TECHNIQUES

The modules for making engineering estimations of the functionality of damaged vehicles are similarly isolated. The related interfaces and file formats for this will also be standardized. This will allow changes to the total damage assessment system with no changes at all to the computer programs involved. Thus, an analyst will be able to develop a new damage assessment list for a particular mission function or environment and test it with a real target simply by specifying the file containing the new list. Should there be a need to change the methodology used for making these engineering approximations, the changes can be incorporated with a minimum of alterations to the program. Only the group of modules that actually deal with the process referred to as "Determine Shot Loss of Utility" in the data flow diagrams will be affected.

6. CONSIDERATION OF TIME-DEPENDENT PHENOMENOLOGY

Much thought has gone into providing the ability to maintain "time awareness" within MUVES. Each threat has a "time stamp" attached to it so that interactions can be causally ordered and so that interaction modules will be able to determine the time of any particular event, should that be necessary. (This information may, of course, be ignored in the interest of efficiency if desired.) This will allow the analyst several capabilities which have been completely unavailable to date. For instance, he will be able to graph the effects of time-delayed damage to a target (including leaks, fires, and gases). He will also be able to combine the effects of multiple hits on a single target. This may be done after the fact, combining the lists of damaged components for separate shots, or it may be done more accurately, using target modifications to change the characteristics of the target before subsequent munition impacts.

VII. FUTURE DIRECTIONS

As mentioned earlier in this report, we have completed the User Survey and the Structured Analysis phase of this project. Our first major implementation goal for MUVES will be to support just the compartment model for threat/target interactions. Because the compartment model is the least complex vulnerability model and therefore the easiest to implement, it will allow us to provide a useful prototype system to analysts as soon as possible. In addition, compartment model analysis is still an important and heavily-used production tool which can benefit greatly from the advantages of the MUVES design.

Our next task in preparing for implementation of this subset of MUVES is to perform the detailed analysis of the interaction modules required for the compartment model. Then we will proceed with the Structured Design, followed by Top-Down Implementation of the overall MUVES system and the family of interaction modules which form the compartment model. In addition, work on the user interface will proceed concurrently with these two phases. Our goal is to have the MUVES version of the compartment model ready for production use by the third quarter of FY88.

Once we have a functional compartment model, we will then use Structured Analysis and Structured Design techniques to define the interaction modules required for a point-burst analysis. In this way we will gradually integrate the various existing component level methodologies into MUVES, including all extant threat/target interaction algorithms.

After we have provided the analytical capabilities available in existing ground vulnerability codes, we will begin work on adding new algorithms and new capabilities to the MUVES environment. In particular, enhanced graphical presentation of assessment results seems very desirable.

REFERENCES

1. C. L. Nail, E. Jackson, and T. E. Beardon, "Vulnerability Analysis Methodology Program (VAMP) — A Combined Compartment-Kill Vulnerability Model", Computer Sciences Corporation Technical Manual CSC TR-79-5585, October 1979.

2. C. L. Nail, "Vulnerability Analysis for Surface Targets (VAST)- An Internal Point-Burst Vulnerability Assessment Model - Revision I", Computer Sciences Corporation Technical Manual CSC TR-82-5740, August 1982.

3. Douglas A. Ringers and F. Tyler Brown, "SLAVE (Simple Lethality and Vulnerability Estimator) Analyst's Guide," US Army Ballistic Research Laboratory Technical Report ARBRL-TR-02333, June 1981.

4. Aivars Ozolins and Paul Tanenbaum, "Assessment of Vulnerability Analysis Capabilities and Requirements," February 1986.

5. E. Yourdon, "Managing the System Life Cycle", Yourdon Press, 1982.

6. T. DeMarco, "Structured Analysis and System Specification", Yourdon Press, 1978.

7. M. Page-Jones, "The Practical Guide to Structured Systems Design", Yourdon Press, 1980.

APPENDIX A

Data Flow Diagrams

MUVES
PROCESS HIERARCHY

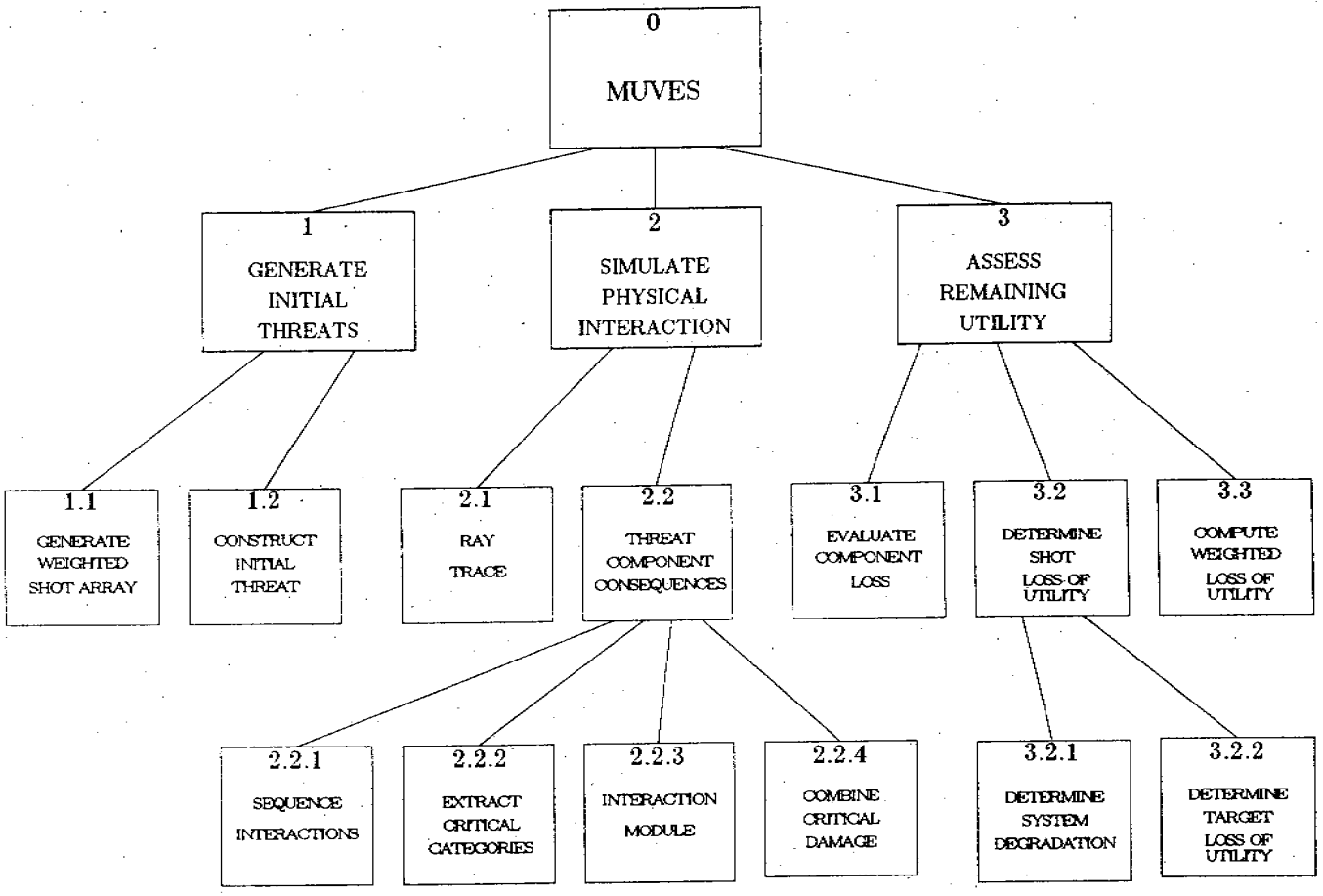

Figure A-1. Tree Structure Showing Hierarchy of MUVES Processes

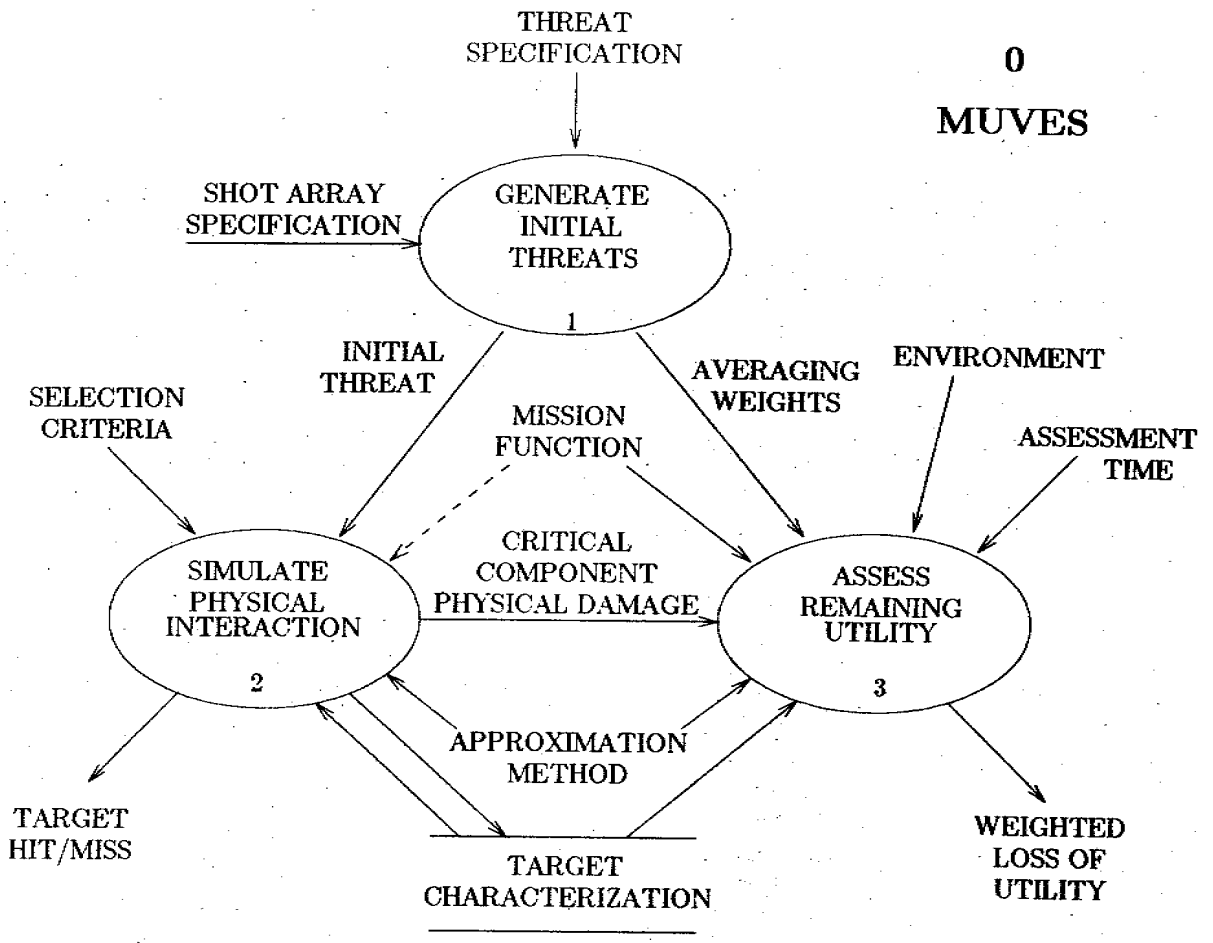

Figure A-2. Top Level Data Flow Diagram of MUVES

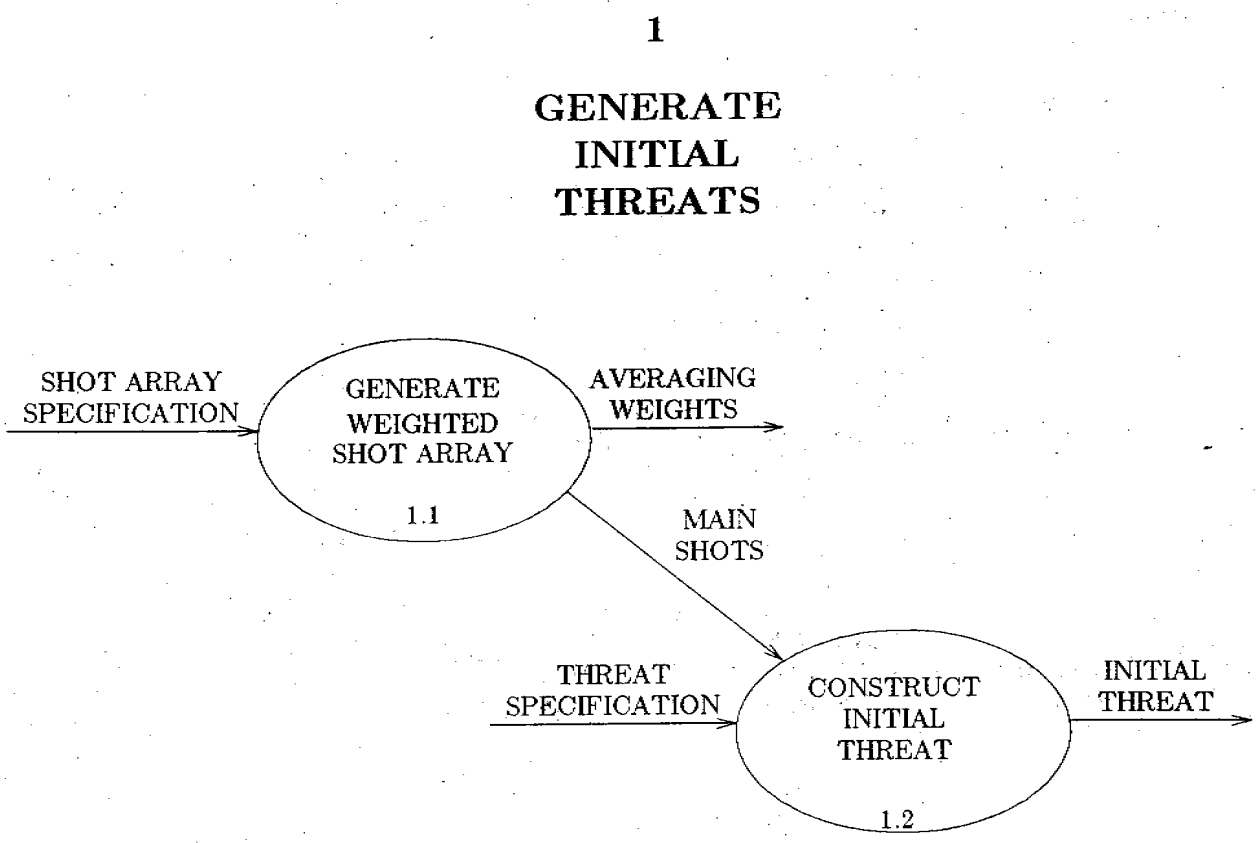

Figure A-3. Data Flow Diagram for Process 1 of MUVES

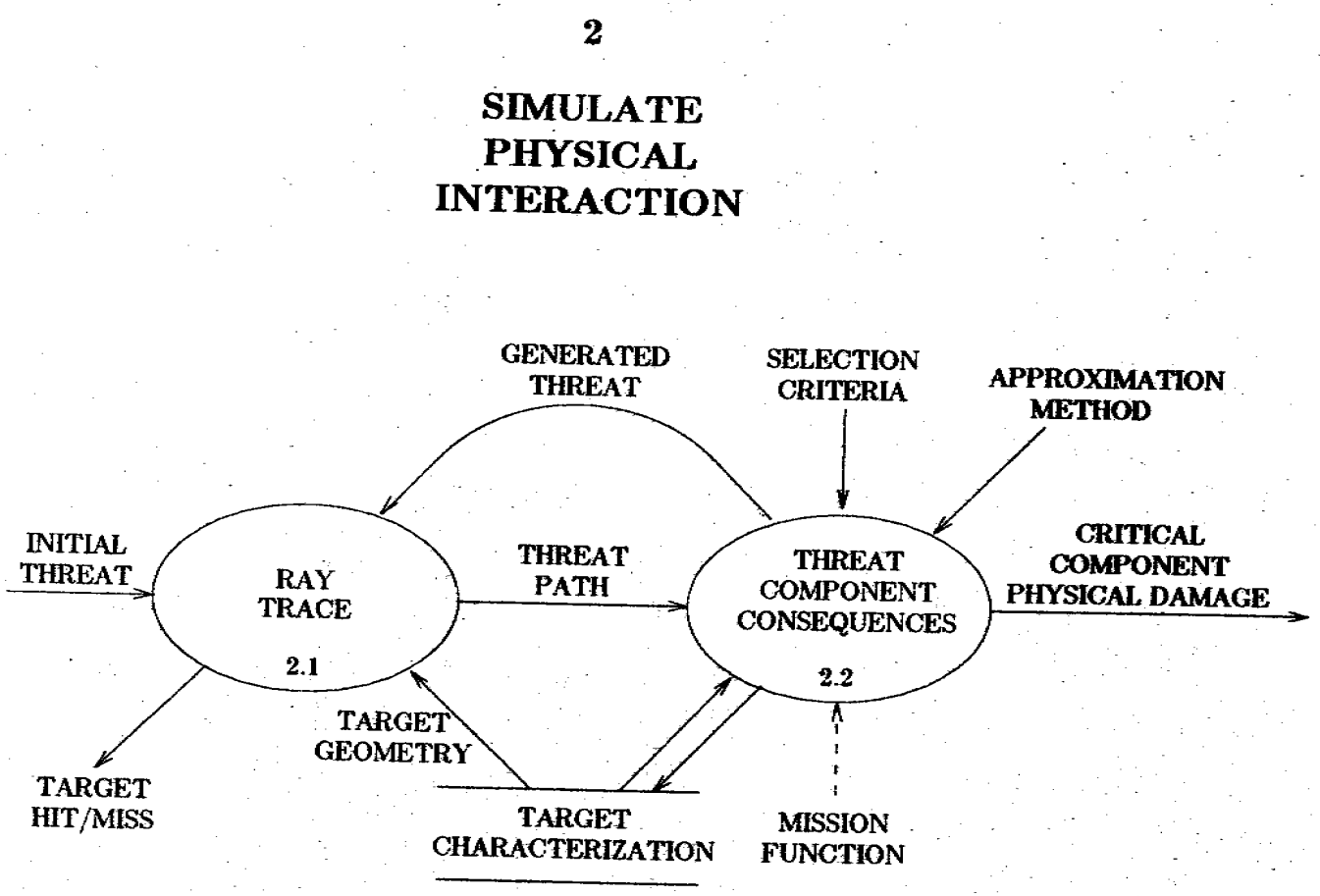

Figure A-4. Data Flow Diagram for Process 2 of MUVES

2.2

THREAT COMPONENT CONSEQUENCES

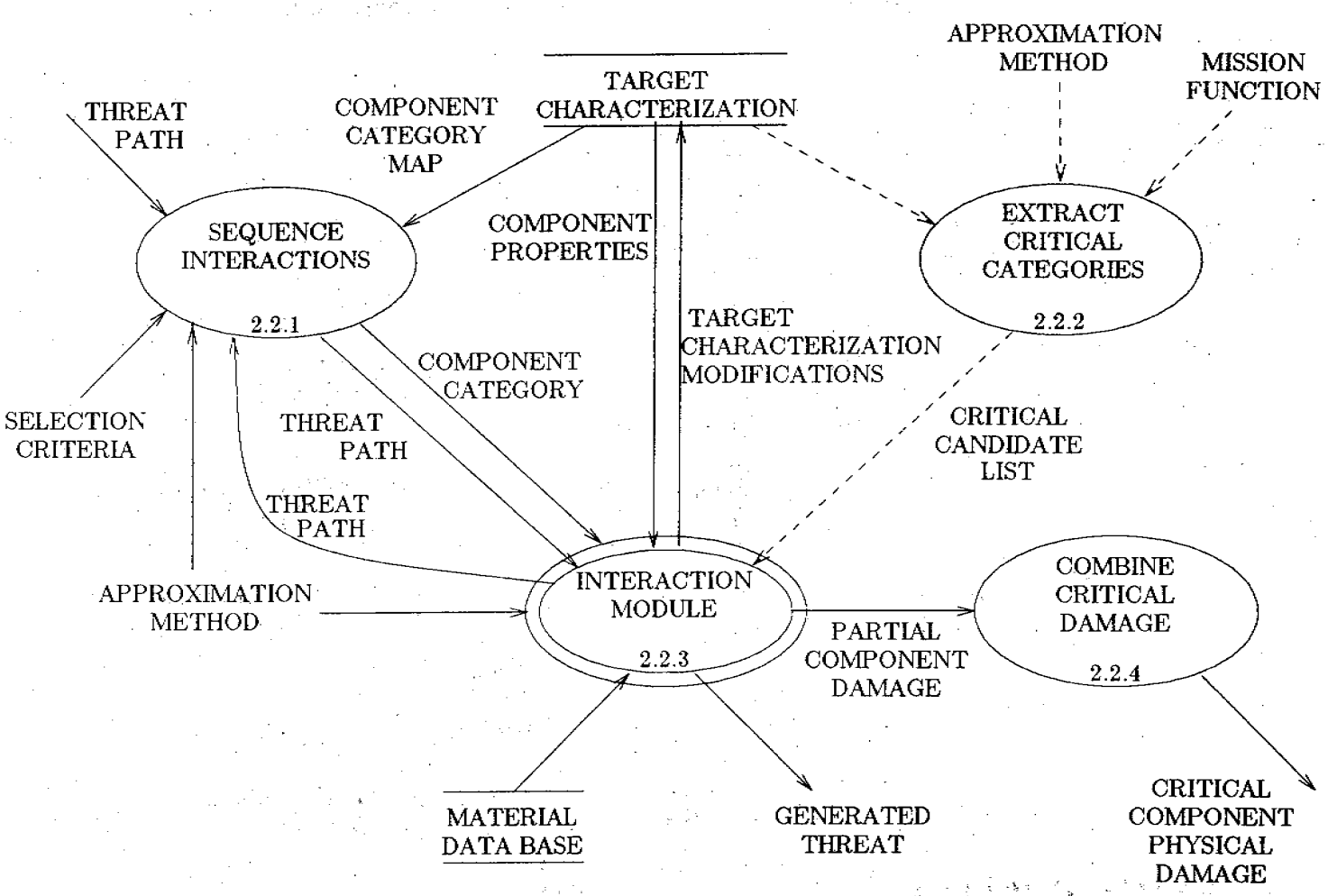

Figure A-5. Data Flow Diagram for Process 2.2 of MUVES

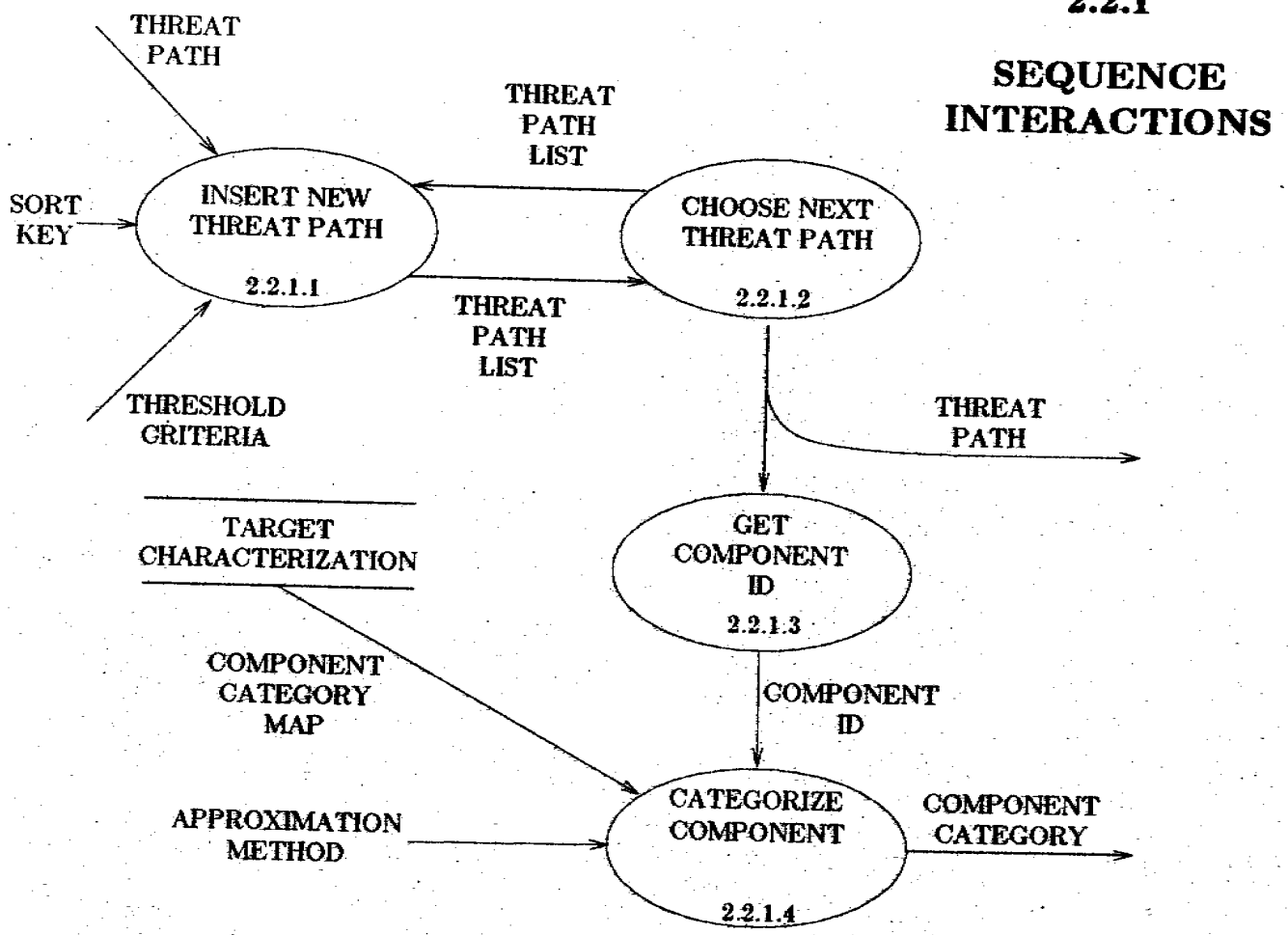

Figure A-6. Data Flow Diagram for Process 2.2.1 of MUVES

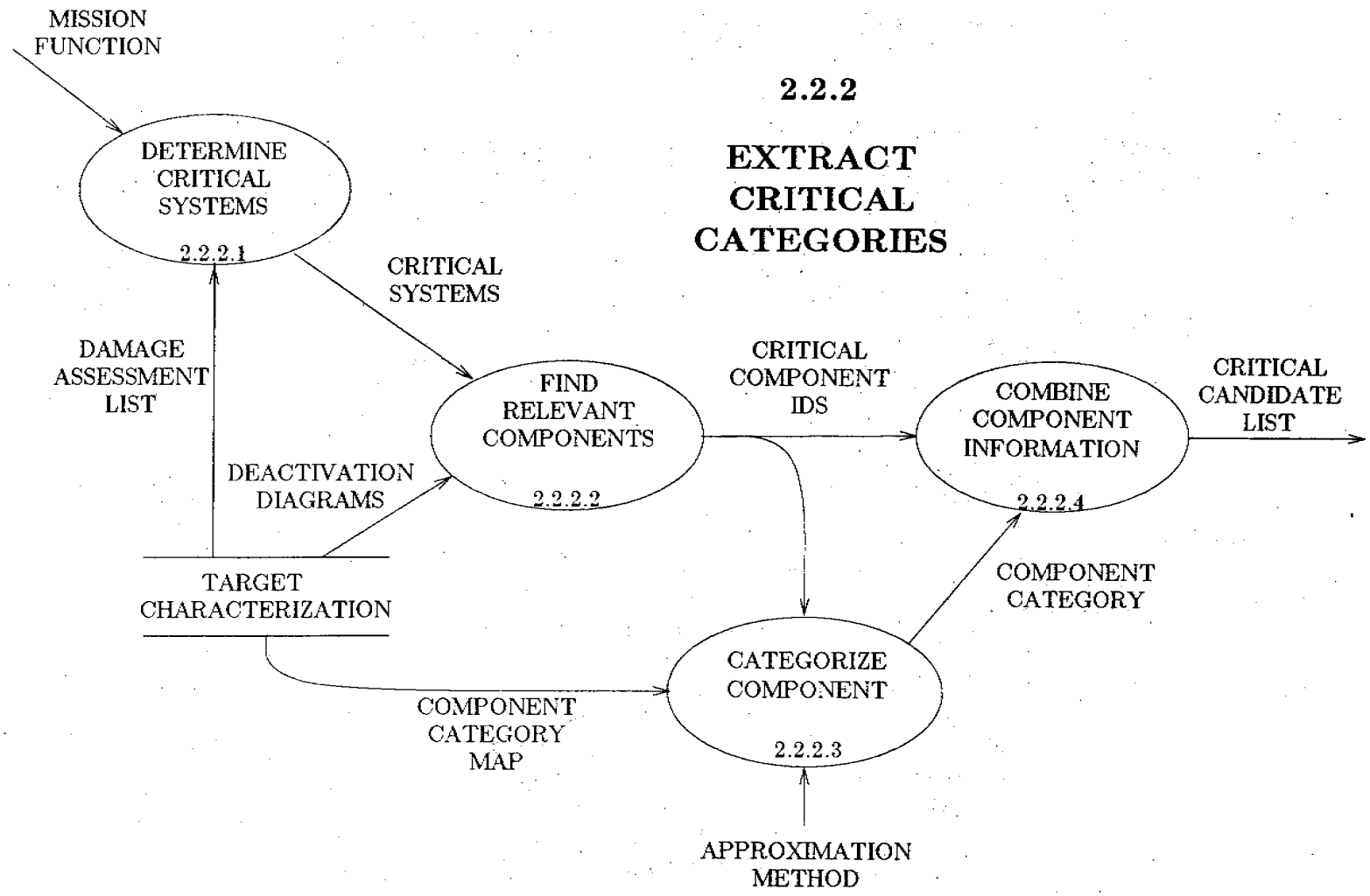

Figure A-7. Data Flow Diagram for Process 2.2.2 of MUVES

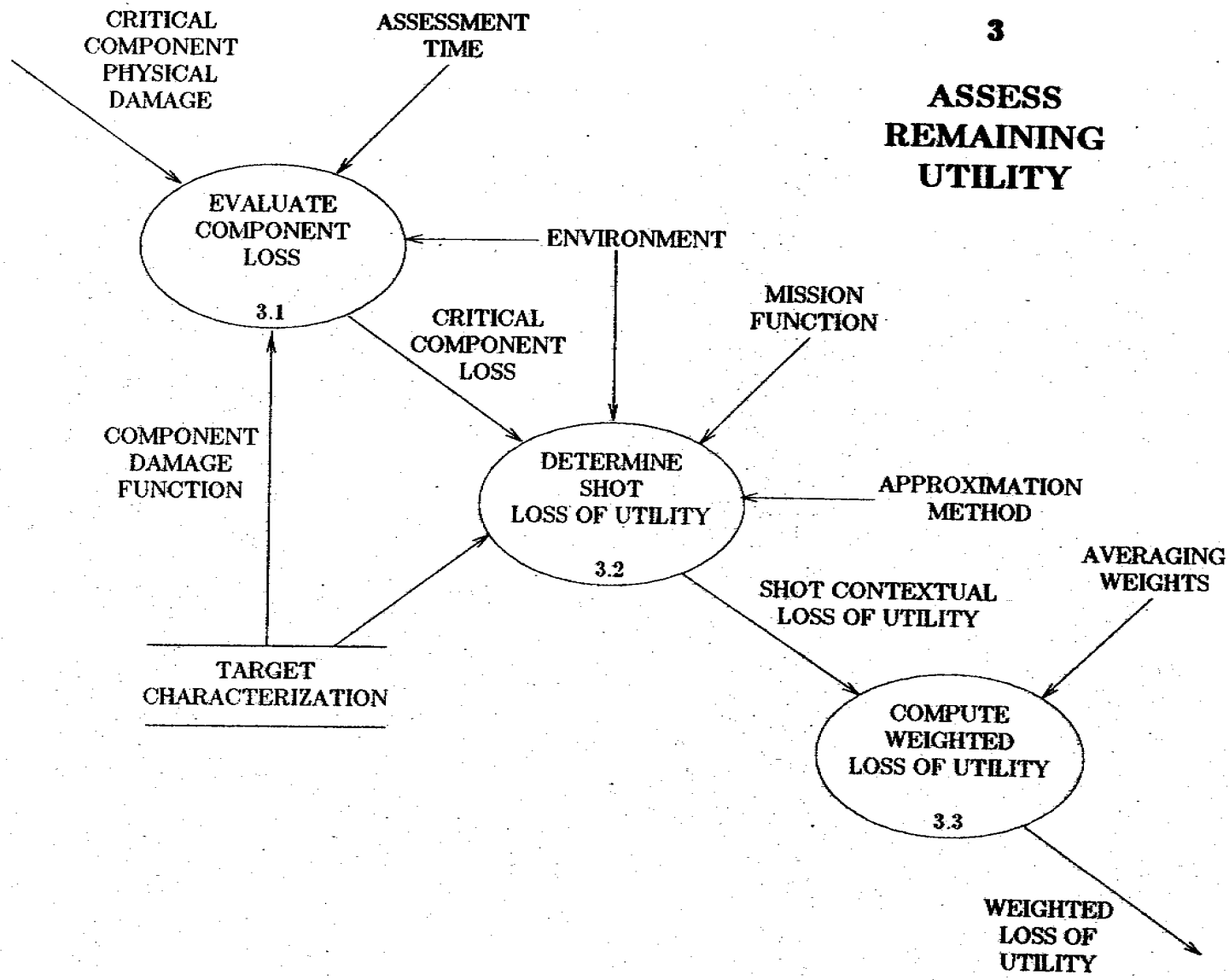

Figure A-8. Data Flow Diagram for Process 3 of MUVES

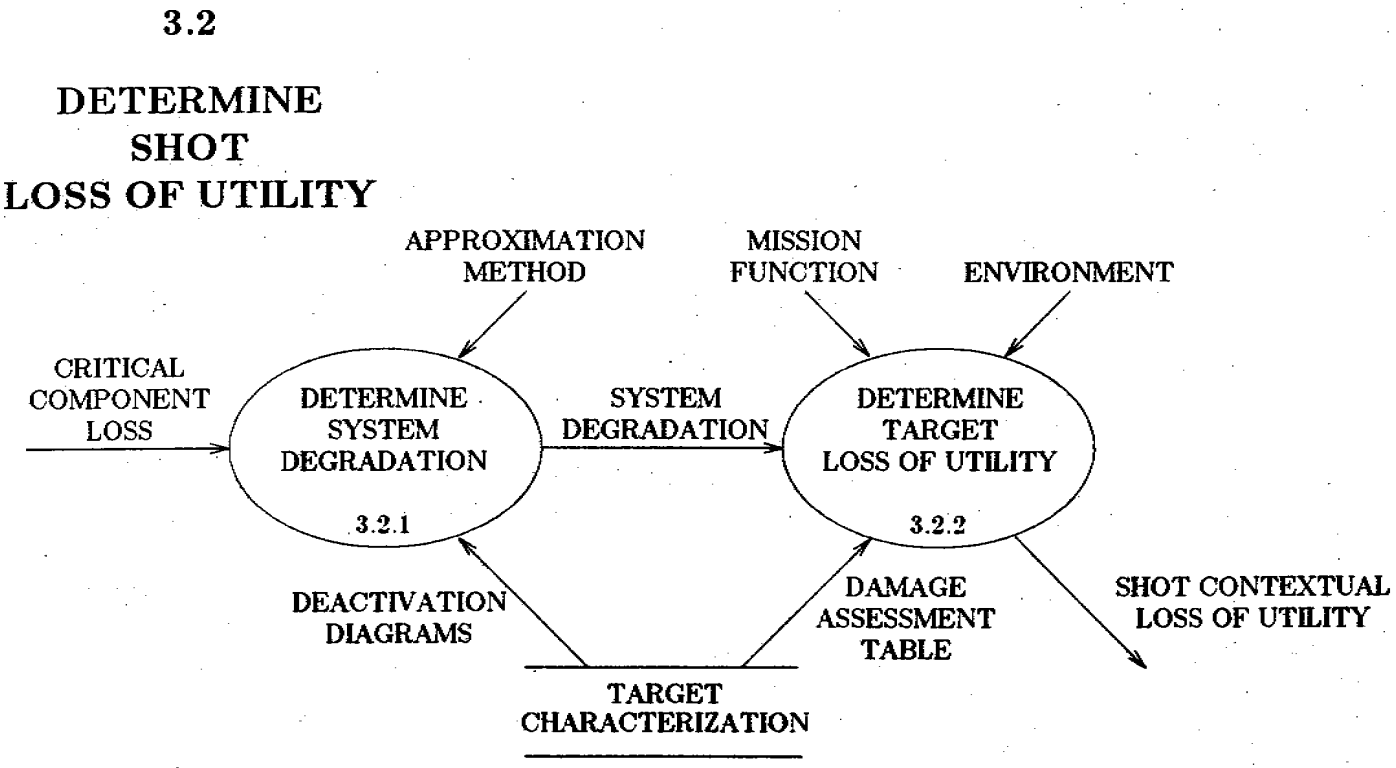

Figure A-9. Data Flow Diagram for Process 3.2 of MUVES

APPENDIX B

Process Definitions

The following notation is used in the Process Definitions in this Appendix:

=	means IS EQUIVALENT TO.
+	means AND.
[ ]	means EITHER - OR; i.e., select one of the options enclosed in the brackets.
{ }	means ITERATIONS OF the component enclosed.
( )	means that the enclosed component is OPTIONAL.

We have used the convention that capitalized words in the data dictionary definitions are terms that are further defined in the data dictionary.

MUVES Processes

ASSESS REMAINING UTILITY	PROCESS 3	Use engineering approximations to estimate the WEIGHTED LOSS OF UTILITY of a TARGET with respect to the MISSION FUNCTION of interest, based on CRITICAL COMPONENT PHYSICAL DAMAGE resulting from physical THREAT/TARGET interaction in the specified ENVIRONMENT, weighted per MAIN SHOT by a set of AVERAGING WEIGHTS.
CATEGORIZE COMPONENT	PROCESS 2.2.1.4 PROCESS 2.2.2.3	Determine COMPONENT CATEGORY for the first COMPONENT in a THREAT PATH, using the appropriate COMPONENT CATEGORY MAP.
CHOOSE NEXT THREAT PATH	PROCESS 2.2.1.2	Select the next remaining THREAT PATH in the THREAT PATH LIST for further processing by the INTERACTION MODULE.
COMBINE COMPONENT INFORMATION	PROCESS 2.2.2.4	Sort COMPONENT CATEGORIES to create a list of unique categories and attach the COMPONENT IDS belonging to those categories.
COMBINE CRITICAL DAMAGE	PROCESS 2.2.4	Sort the collection of PARTIAL COMPONENT DAMAGES to form a list of physical damage states for each CRITICAL COMPONENT damaged by the INITIAL THREAT or any GENERATED THREATS.
COMPUTE WEIGHTED LOSS OF UTILITY	PROCESS 3.3	Combine the individual SHOT CONTEXTUAL LOSSES OF UTILITY according to some weighting function for MAIN SHOTS represented by AVERAGING WEIGHTS to produce a single WEIGHTED LOSS OF UTILITY for the given collection of INITIAL THREATS.
CONSTRUCT INITIAL THREAT	PROCESS 1.2	Combine THREAT SPECIFICATION for the desired THREAT TYPE with a MAIN SHOT to produce an INITIAL THREAT.

DETERMINE CRITICAL SYSTEMS	PROCESS 2.2.2.1	Examine DAMAGE ASSESSMENT TABLE to determine which TARGET subsystems are necessary for the TARGET to perform the desired MISSION FUNCTION.
DETERMINE SHOT LOSS OF UTILITY	PROCESS 3.2	Determine the TARGET'S loss of ability to perform the desired MISSION FUNCTION as a result of CRITICAL COMPONENT PHYSICAL DAMAGE resulting from THREAT/TARGET interaction for a particular MAIN SHOT in the context of the specified ENVIRONMENT.
DETERMINE SYSTEM DEGRADATION	PROCESS 3.2.1	Use DEACTIVATION DIAGRAMS appropriate to the desired APPROXIMATION METHOD to combine CRITICAL COMPONENT LOSSES into loss of function of a subsystem within the TARGET.
DETERMINE TARGET LOSS OF UTILITY	PROCESS 3.2.2	Combine losses of function of TARGET subsystems into loss of ability to perform MISSION FUNCTION for a single MAIN SHOT based on the DAMAGE ASSESSMENT TABLE appropriate to the specified ENVIRONMENT and the desired MISSION FUNCTION.
EVALUATE COMPONENT LOSS	PROCESS 3.1	Estimate the probable loss of a COMPONENT at the ASSESSMENT TIME given the aggregate physical damage to that COMPONENT and information specific to the particular ENVIRONMENT.
EXTRACT CRITICAL CATEGORIES	PROCESS 2.2.2	Examine the TARGET'S structural information (*i.e.*, DAMAGE ASSESSMENT TABLES and DEACTIVATION DIAGRAMS for the desired APPROXIMATION METHOD) for the MISSION FUNCTION of interest to produce the CRITICAL CANDIDATE LIST required to ASSESS REMAINING UTILITY.
FIND RELEVANT COMPONENTS	PROCESS 2.2.2.2	Examine the DEACTIVATION DIAGRAMS belonging to each CRITICAL SYSTEM in order to find the COMPONENTS needed for that system to continue to function.

GENERATE INITIAL THREATS	PROCESS 1	For each MAIN SHOT resulting from the SHOT ARRAY SPECIFICATION generate an AVERAGING WEIGHT and one or more INITIAL THREATS according to the THREAT SPECIFICATION.
GENERATE WEIGHTED SHOT ARRAY	PROCESS 1.1	Generate start points and directions, as well as an AVERAGING WEIGHT to simulate probable shot dispersion, for each MAIN SHOT specified by the SHOT ARRAY SPECIFICATION.
GET COMPONENT ID	PROCESS 2.2.1.3	Find the COMPONENT ID for the first COMPONENT in a particular THREAT PATH.
INSERT NEW THREAT PATH	PROCESS 2.2.1.1	Filter new THREATS according to the chosen THRESHOLD CRITERIA and insert the THREAT PATH into the proper location in the THREAT PATH LIST as specified by the desired SORT KEY.
INTERACTION MODULE	PROCESS 2.2.3	Determine consequences of the physical interaction between a specific THREAT and a specific COMPONENT according to a specified APPROXIMATION METHOD, resulting in any or all of: undeflected residual THREAT PATH, GENERATED THREATS such as deflected and degraded original THREAT or subsidiary THREATS (*e.g.*, spall) [this module will have the responsibility of including the THREAT STAGE in all GENERATED THREATS], CRITICAL COMPONENT PHYSICAL DAMAGE, and possibly TARGET CHARACTERIZATION MODIFICATIONS. (There are actually many individual modules conforming to a single common program interface, each capable of handling all interactions of at least one combination of THREAT TYPE, COMPONENT CATEGORY, and APPROXIMATION METHOD, and new or experimental modules may be supplied by analysts without affecting the structure of the system.)

RAY TRACE	PROCESS 2.1	Determine which COMPONENTS of the TARGET intersect the THREAT VECTOR of an INCIDENT THREAT, and calculate geometric information pertinent to each intersection.
SEQUENCE INTERACTIONS	PROCESS 2.2.1	Arrange THREAT PATHS in the correct order, according to the chosen SORT KEY, then route them one-by-one to one of an array of INTERACTION MODULES selected on the basis of APPROXIMATION METHOD, THREAT TYPE, and COMPONENT CATEGORY for the first remaining COMPONENT in each THREAT PATH.
SIMULATE PHYSICAL INTERACTION	PROCESS 2	Determine the types and extent of CRITICAL COMPONENT PHYSICAL DAMAGE as a result of the interaction of the given INITIAL THREAT and all GENERATED THREATS with the TARGET, analyzed in detail appropriate to the APPROXIMATION METHOD.
THREAT COMPONENT CONSEQUENCES	PROCESS 2.2	Evaluate the consequences of a THREAT PATH traversing the TARGET according to the specified APPROXIMATION METHOD and COMPONENT CHARACTERISTICS, producing information about CRITICAL COMPONENT PHYSICAL DAMAGE, secondary GENERATED THREATS (*e.g.*, spall representative rays), and possibly TARGET CHARACTERIZATION MODIFICATIONS.

APPENDIX C

Data Flow Definitions

The following notation is used in the Data Flow Definitions in this Appendix:

=	means IS EQUIVALENT TO.
+	means AND.
[ ]	means EITHER - OR; i.e., select one of the options enclosed in the brackets.
{ }	means ITERATIONS OF the component enclosed.
( )	means that the enclosed component is OPTIONAL.

We have used the convention that capitalized words in the data dictionary definitions are terms that are further defined in the data dictionary.

MUVES Data Flow Definitions

APPROXIMATION METHOD

specific choice of an integrated package of THREAT/COMPONENT INTERACTION MODULES appropriate to a particular method of simplifying the analysis of THREAT/TARGET interaction (*e.g.*, compartment model, simplified point burst, detailed point burst)

ASSESSMENT TIME

time or times at which the analyst wishes to evaluate the functionality of the damaged TARGET

AVERAGING WEIGHTS

weight factors for each MAIN SHOT used to combine the SHOT CONTEXTUAL LOSSES OF UTILITY resulting from INITIAL THREATS into a single WEIGHTED LOSS OF UTILITY distributed over the TARGET

COMPONENT

piece of a TARGET, having definite geometry, which may interact as a single unit with a THREAT, according to the APPROXIMATION METHOD (*e.g.*, armor plate, road wheel, fuel line, crew compartment) [may be a combination of smaller COMPONENTS, depending on the desired level of modeling detail]

COMPONENT CATEGORY

classification of a particular COMPONENT into one of several standard types (*e.g.*, RHA, personnel)

COMPONENT CATEGORY MAP

map of COMPONENT IDS to generic COMPONENT CATEGORIES

COMPONENT PROPERTIES

all information about a COMPONENT which may affect the physical interaction with a particular THREAT and which may be relevant to damage mechanisms

COMPONENT DAMAGE FUNCTION

information required to determine CRITICAL COMPONENT LOSS given CRITICAL COMPONENT PHYSICAL DAMAGE

COMPONENT ID

label that uniquely identifies a specific COMPONENT within a TARGET

COMPONENT TRACE

COMPONENT ID +
TRACE through COMPONENT +
COMPONENT surface geometry

CRITICAL CANDIDATE LIST	list of unique COMPONENTS which qualify as CRITICAL COMPONENTS and a list of unique corresponding COMPONENT CATEGORIES: {COMPONENT CATEGORY + {COMPONENT ID} }
CRITICAL COMPONENT	COMPONENT which is part of any TARGET subsystem (appears at least once in that subsystem's DEACTIVATION DIAGRAMS) needed for continued performance of the selected MISSION FUNCTION(S) in the selected ENVIRONMENT
CRITICAL SYSTEM	any TARGET subsystem needed for continued performance of the desired MISSION FUNCTION(S) in the selected ENVIRONMENT
CRITICAL COMPONENT LOSS	numeric estimation of the extent of damage done to a CRITICAL COMPONENT (may be more than one number and exact meaning depends on method used to DETERMINE SHOT LOSS OF UTILITY, but usually probability of a hard kill of the CRITICAL COMPONENT)
CRITICAL COMPONENT PHYSICAL DAMAGE	*physical* characterization of the damage to all CRITICAL COMPONENTS after the INITIAL THREAT and all GENERATED THREATS for a particular MAIN SHOT have interacted with the entire TARGET: { COMPONENT ID + COMPONENT CATEGORY + {damage type + DAMAGE VARIABLES} }
DAMAGE ASSESSMENT TABLE	list of contributions of combined SYSTEM DEGRADATIONS to SHOT CONTEXTUAL LOSS OF UTILITY in the context of a specific MISSION FUNCTION and ENVIRONMENT
DAMAGE VARIABLES	relevant parameters to physically quantify a particular type of damage to a COMPONENT and to determine the time at which that damage occurs (*e.g.*, hole size, number of impacting fragments, transient overpressure)
DEACTIVATION DIAGRAM	description of TARGET subsystem operation in terms of operation of CRITICAL COMPONENTS

ENVIRONMENT	military situation within which THREAT/TARGET interaction is assumed to occur (*e.g.*, nighttime in forest, rough roads, "typical")
GENERATED THREAT	INCIDENT THREAT (other than an undeflected residual penetrator) which is produced by the interaction of a THREAT with some COMPONENT (*e.g.*, deflected residual penetrator, spall representative ray)
INCIDENT THREAT	information concerning any THREAT to be traced through a TARGET: THREAT TYPE + THREAT STAGE + THREAT PARAMETERS + THREAT VECTOR
INITIAL THREAT	INCIDENT THREAT specified as present before any interaction with the TARGET has occurred
MAIN SHOT	one of an array of assumed trajectories for INITIAL THREATS: origin point of THREAT + direction in which THREAT is moving before encounter with TARGET
MISSION FUNCTION	aspect of military capabilities under consideration (*e.g.*, mobility, speed, firepower, personnel loss, cost/time to repair)
PARTIAL COMPONENT DAMAGE	*physical* characterization of the damage to a CRITICAL COMPONENT after a particular INCIDENT THREAT has interacted with it: COMPONENT ID + COMPONENT CATEGORY + {damage type + DAMAGE VARIABLES}
SELECTION CRITERIA	THRESHOLD CRITERIA + SORT KEY
SHOT ARRAY SPECIFICATION	parameters necessary to generate the pattern of MAIN SHOTS desired by the analyst to emulate the dispersion of anticipated trajectories

SHOT CONTEXTUAL LOSS OF UTILITY	estimated loss of TARGET'S ability to perform the desired MISSION FUNCTION in a particular ENVIRONMENT at some ASSESSMENT TIME as a result of interaction with one or more INITIAL THREATS
SORT KEY	specification, usually implicit in the choice of APPROXIMATION METHOD, used to determine the order in which individual THREAT PATHS will be processed by the INTERACTION MODULES (e.g. time-ordered, "first-come-first-served", THREAT STAGE)
SYSTEM DEGRADATION	estimated loss of function of TARGET subsystem (usually probability of total loss) for a given MAIN SHOT at the desired ASSESSMENT TIME (exact meaning depends on package design of CRITICAL COMPONENT LOSS and DEACTIVATION DIAGRAMS)
TARGET	system of interrelated COMPONENTS which performs one or more MISSION FUNCTIONS of interest to the analyst
TARGET CHARACTERIZATION MODIFICATIONS	changes in TARGET properties or structure (other than CRITICAL COMPONENT PHYSICAL DAMAGE) resulting from interaction with a THREAT (*e.g.*, deformation, degradation of materials) which must be considered for accurate interaction with subsequent INITIAL THREATS
TARGET GEOMETRY	MGED description of COMPONENT geometries for an entire TARGET
TARGET HIT/MISS	boolean value indicating whether a particular INITIAL THREAT hit any portion of the TARGET, or missed it altogether
THREAT	phenomenon capable of damaging a TARGET as a result of physical interaction (*e.g.*, kinetic energy projectile, fragment, blast wave, energy beam)
THREAT PARAMETERS	characterization of the state of a THREAT (exact contents and interpretation depend on THREAT TYPE)

THREAT PATH

specific THREAT with a projected TRACE through the TARGET:
THREAT TYPE +
THREAT PARAMETERS +
THREAT VECTOR +
{COMPONENT TRACE}

THREAT PATH LIST

a list of THREAT PATHS ordered by a particular SORT KEY

THREAT RELATIVE VELOCITY

velocity of actual impending objects (*e.g.*, fragment, explosively formed penetrator, etc.), relative to the incoming munition which produced them

THREAT SPECIFICATION

properties of a THREAT as it exists before any interaction with a TARGET
{THREAT TYPE +
THREAT STAGE +
THREAT PARAMETERS +
THREAT RELATIVE VELOCITY +
THREAT START TIME}

THREAT STAGE

integer variable to indicate the level of a particular THREAT in the hierarchy of creation dependencies (*e.g.*, a shaped-charge (INITIAL THREAT -- stage 1) impacts armor and produces spall (GENERATED THREAT -- stage 2))

THREAT START TIME

time at which the THREAT is fired

THREAT TYPE

generic classification of a THREAT (*e.g.*, shaped charge jet, spall representative ray)

THREAT VECTOR

origin point of THREAT +
time at origin point +
velocity vector of THREAT

THRESHOLD CRITERIA

criteria, usually implicit in the choice of APPROXIMATION METHOD, used to determine whether a particular THREAT PATH should receive further consideration by the INTERACTION MODULES

TRACE	origin of a ray + vector from the origin to the end point of the ray
WEIGHTED LOSS OF UTILITY	weighted average of SHOT CONTEXTUAL LOSSES OF UTILITY, combined using the AVERAGING WEIGHTS

DISTRIBUTION LIST

No. of Copies	Organization
12	Administrator Defense Technical Information Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
10	Central Intelligence Agency Dissemination Branch Room GE-47 HQS Washington, DC 20502
1	Defense Intelligence Agency Pentagon ATTN: Herb Dimick Washington, DC 20301
1	HQDA ATTN: DAMA-ART-M Washington, DC 20310
1	HQDA ATTN: DAMA-ARZ-B (COL Ken Evans) Washington, DC 20310
1	HQDA Dr. Louise Cameron Director for Research & Technology SARD-TR, The Pentagon Washington, DC 20310-0600
1	HQDA Asst Chief of Staff for Intelligence Chief, Space Systems Division ATTN: Joseph Varnadore Washington, DC 20310-1067
1	HQDA Hunter M. Woodall, Jr. Director, Program Analysis Pentagon, Rm 3E360 Washington, DC 20310-0103
9	Defense Advanced Research Projects Agency ATTN: Mr. B. Bandy Dr. R. Kahn Dr. C. Kelly Mr. P. Losleben Dr. J. Lupo Mr. F. Patten Dr. Reynolds Mr. S. Squires COL J. Thorpe 1400 Wilson Boulevard Arlington, VA 22209
1	Central Intelligence Agency ATTN: ORD/IERD (J. Fleisher) Washington, DC 20505
1	Col Robert Gomez/OST P.O. Box 1925 Washington, DC 20505
1	Headquarters Army Materiel Command ATTN: AMCDRA-PD (F. Michel) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander US Army Material Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander US Army Materiel Command ATTN: AMCMT (John Kicak) 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: AMCDE-PM (Dan Marks) 5001 Eisenhower Avenue Alexandria, VA 22333

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Materiel Command ATTN: AMCTD-PT (Alan Elkins) 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: AMCPD (Darold Griffin) 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: AMCPD-PM (Jim Sullivan) 5001 Eisenhower Avenue Alexandria, VA 22333
2	Commander US Army Materiel Command ATTN: AMCPM-LOTA (Robert Hall) (MAJ C. Purdin) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander US Army Armament Research and Development Center ATTN: SMCAR-FSS-E (Jack Brooks) Dover, NJ 07801
1	Commander US Army Armament Research and Development Center ATTN: SMCAR-TD (Jim Killen) Dover, NJ 07801
1	Commander US Army Armament Research and Development Center ATTN: SMCAR-TDC Dover, NJ 07801
1	Commander US Army Armament Research and Development Center ATTN: SMCAR-TSS Dover, NJ 07801
4	Commander US Army Aviation Research and Development Command ATTN: AMSAV-E AMSAV-GT (R. Lewis) AMSAV-NC (H. Law) (S. Meyer) 4300 Goodfellow Blvd St. Louis, MO 63120
1	Commander Belvoir Research, Development and Engineering Center ATTN: STRBE-JDA (Melvin Goss) Fort Belvoir, VA 22060-5606
1	Commander Belvoir Research, Development and Engineering Center ATTN: STRBE-FC (Ash Patil) Fort Belvoir, VA 22060-5606
1	Director Benet Weapons Laboratory Armament Research, Development and Engineering Center ATTN: SMCAR-LDB-TL Watervliet, NY 12189-4050
1	Commander US Army Armament, Munitions and Chemical Command ATTN: SMCAR-ESP-L Rock Island, IL 61299
1	Director US Army Air Mobility Research and Development Laboratory Ames Research Center Moffet Field, CA 94035
1	Commander US Army Communication Electronics Command ATTN: AMSEL-ED Fort Monmouth, NJ 07703-5301

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Electronics Research & Development Command Technical Library ATTN: DELSD-L (Reports Section) Fort Monmouth, NJ 07703-5301
3	Commander US Army Electronics Research & Development Command Night Vision & Electronic Optics Lab ATTN: DELMV-L (Dr. R. Buser) AMSEL-RD-MV-V (John Ho) AMSEL-NV-V (John Palmer) Fort Belvoir, VA 22060-5677
4	Commander US Army Foreign Science and Technology Center ATTN: AMXST-CS-1 (T. Walker) (G. Hargis) (S. Eitleman) (R. Witnebal) 220 Seventh St, N.E. Charlottesville, VA 22901
1	Commander US Army Foreign Science & Technology Center ATTN: AIAST-OC (Mr. Donald Dinger) 220 Seventh Street, NE Charlottesville, VA 22901-5396
1	Commander US Army Foreign Science & Technology Center ATTN: AIFRCC (Richard Comfort) 220 Seventh Street, NE Charlottesville, VA 22901-5396
3	Commander US Army Foreign Science & Technology Center ATTN: AIFRS (Dr. Gordon Spencer) (Mr. John McKay) (Mr. C. Grobmyer) 220 Seventh Street, NE Charlottesville, VA 22901-5396
1	Commander US Army Foreign Science & Technology Center ATTN: DRXST-SD4 (T. D'Isepo) Charlottesville, VA 22901
1	US Army Harry Diamond Laboratories ATTN: DRXCM (Mr. Halsey) 2800 Powdermill Road Adelphi, MD 20783-1197
1	US Army Harry Diamond Laboratories ATTN: SLCHD-RT (Peter Johnson) 2800 Powdermill Road Adelphi, MD 20783-1197
1	Commander US Army INSCOM ATTN: IAOPS-SE-M (George Maxfield) Arlington Hall Station Arlington, VA 22212-5000
3	Commander US Army Missile Command ATTN: AMSMI-RD AMSMI-RD-GC-T (R. Alongi) AMSMI-RGT (J. Bradas) Redstone Arsenal, AL 35898
2	Commander US Army Missile Command ATTN: DRSMI-REX (W. Pittman) DRSMI-YRT (Pete Kirkland) Redstone Arsenal, AL 35898

DISTRIBUTION LIST

No. of Copies	Organization
2	Commander US Army Tank-Automotive Command ATTN: AMSTA-TSL AMSTA-ZS Warren, MI 48397-5000
3	Commander US Army Tank-Automotive Command ATTN: AMSTA-RSC (John Bennett) (Walt Wynbelt) (Wally Mick) Warren, MI 48397-5000
4	Commander US Army Tank-Automotive Command ATTN: AMSTA-ZSS (J. Thompson) (O. Renius) (D. Reese) (J. Soltez) Warren, MI 48397-5000
1	Commander US Army Tank-Automotive Command ATTN: AMSTA-NKS (D. Cyaye) Warren, MI 48397-5000
2	Commander US Army Tank Automotive Command ATTN: AMSTA-RGE (Dr. R. Munt) (R. McClelland) Warren, MI 48397-5000
1	Director US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL White Sands Missile Range, NM 88002-5022
1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905
1	Commander US Army Development and Employment Agency ATTN: MODE-TED-SAB Fort Lews, WA 98433-5000
1	Commander US Army Vulnerability Assessment Laboratory ATTN: SLCVA-CF (Gil Apodaca) White Sands Missile Range, NM 88002-5513
1	Director US Army Cold Regions Research & Development Laboratory ATTN: Tech Dir (Lewis Link) 72 Lyme Road Hanover, NH 03755
1	US Army Corps of Engineers Assistant Director Research & Development Directorate (Military Programs) ATTN: Mr. B. Benn 20 Massachusetts Avenue, N.W. Washington, DC 20314-1000
2	US General Accounting Office Program Evaluation and Methodology Division ATTN: Robert G. Orwin Joseph Sonnefeld Room 5844 441 G Street, NW Washington, DC 20548
1	Director US Army Industrial Base Engineering Activity ATTN: AMXIB-MT Rock Island, IL 61299-7260
1	Director US Army Industrial Base Engineering Activity ATTN: AMXIB-PS (Steve McGlone) Rock Island, IL 61299-7260

DISTRIBUTION LIST

No. of Copies	Organization
3	Commander and Director US Army Engineer Waterways Experiment Station ATTN: WESEN (Dr. V. LaGarde) (Mr. W. Grabau) WESEN-C (Mr. David Meeker) P.O. Box 631 Vicksburg, MS 39180-0631
1	Technical Director US Army Engineer Topographic Laboratories ATTN: Mr. Walter E. Boge Fort Belvoir, VA 22060-5546
1	Los Alamos National Laboratories ATTN: MS-F600, Gary Tietgen P.O. Box 1663 Los Alamos, NM 87545
1	Sandia National Laboratories Division 1623 ATTN: Larry Hostetler Albuquerque, NM 87185
1	Sandia National Laboratories Division 1611 ATTN: Tom James Albuquerque, NM 87185
1	Commander Naval Material Command ATTN: F. Gale Washington, DC 20360
1	Naval Intelligence Command ATTN: NIPSSA-333 (Paul Fessler) 4600 Silver Hill Road Washington, DC 20389
1	Commander Intelligence Threat Analysis Center ATTN: Bill Davies Washington Navy Yard Bldg 203 (Stop 314) Washington, DC 20374-2136
2	Intelligence Threat Analysis Center Intell Image Prod Div ATTN: John Creighton Al Fuerst Washington Navy Yard Bldg 213 (IAX-O-II) Washington, DC 20310
1	Commander David W. Taylor Naval Ship & Development Center ATTN: J. Schot Bethesda, MD 20084
1	Naval Weapons Center ATTN: Code 39104 M. H. Keith China Lake, CA 93555
2	AFWAL/MLTC ATTN: LT Robert Carringer Dave Judson Wright-Patterson AFB, OH 45433-6533
1	AFWAL/AARF ATTN: CPT John Poachon Wright-Patterson AFB, OH 45433-6533
1	ASD/XRJ ATTN: Ed Mahen Wright-Patterson AFB, OH 45433
1	AD/CZL ATTN: James M. Heard Eglin AFB, FL 32542-5000
1	AD/ENYW ATTN: Jim Richardson Eglin AFB, FL 32542-5000
1	IDA ATTN: Dr. Lowell Tonnessen 1801 N. Beauregard Street Alexandria, VA 22311

DISTRIBUTION LIST

No. of Copies	Organization
1	Department of Commerce National Bureau of Standards Manufacturing Systems Group ATTN: B. Smith Washington, DC 20234
1	A O Smith Data Systems Division ATTN: H. Vickerman 8901 North Kildeer Court Brown Deer, WI 53209
1	Alliant Computer Company ATTN: David Micciche 1 Monarch Drive Littleton, MA 01460
2	Applicon Incorporated ATTN: J. Horgan M. Schussel 32 Second Avenue Burlington, MA 01803
1	Boeing Corporation ATTN: Mail Stop 48-88 (Wayne Hammond) P.O. Box 3707 Seattle, WA 98124
1	Computer Sciences Corp 200 Sparkman Drive Huntsville, AL 35805
3	Computervision Corporation ATTN: A. Bhide V. Geisberg R. Hillyard 201 Burlington Road Bedford, MA 01730
1	Decision Science Consortium, Inc. ATTN: Mr. T. Bresnick Suite 421 7700 Leesburg Pike Falls Church, VA 22043
1	Denver Research Institute ATTN: Mr. Larry G. Ullyatt P. O. Box 10127 Denver, CO 80210
1	Dialog Information Services ATTN: Mr. R. Reklis 3460 Fillview Blvd Palo Alto, CA 94304
1	Environmental Research Institute of Michigan ATTN: Mr. Kozma P.O. Box 8618 Ann Arbor, MI 48107
1	E-OIR Measurements, Inc. ATTN: Russ Moulton P.O. Box 3348 College Station Fredericksburg, VA 22402
6	FMC Corporation Ordnance Engineering Division ATTN: M. Hatcher L. House J. Jackson M. Krull E. Maddox R. Musante 1105 Coleman Ave, Box 1201 San Jose, CA 95108
1	FMC Corporation Northern Ordnance Division ATTN: M311, Barry Brown 4800 East River Road Minneapolis, MN 55421
1	John Fluke Mfg Co, Inc. ATTN: D. Gunderson P.O. Box C9090 Everett, WA 98206
1	General Dynamics Data Systems Services ATTN: R. Fridshal P.O. Box 80847 San Diego, CA 92138

DISTRIBUTION LIST

No. of Copies	Organization
3	General Motors Corporation Research Laboratories ATTN: J. Boyse J. Joyce R. Sarraga Warren, MI 48090
1	Gettysburg College Box 405 Gettysburg, PA 17325
1	Global Analytics, Inc. ATTN: Mr. Gaelen R. Daum 10065 Old Grove Road San Diego, CA 92131
1	GTRI-RAIL-MAD ATTN: Mr. Joe Bradley CRB 577 Atlanta, GA 30332
1	Jet Propulsion Laboratory California Institute of Technology ATTN: D. Lewis 4800 Oak Grove Drive Pasadena, CA 91109
1	Keweenaw Research Center Michigan Technological University ATTN: Bill Reynolds Houghton, MI 49931
3	Lincoln Laboratory MIT Surveillance Systems Group ATTN: R. Barnes G. Knittel J. Kong 244 Wood Street Lexington, MA 02173-0073
3	Lockheed-California Company ATTN: C. A. Burton R. J. Ricci M. Steinberg Burbank, CA 91520
1	Lockheed-Georgia Company ATTN: J. Tulkoff Marietta, GA 30063
1	LTV ATTN: Mike Logan P.O. Box 225907 Mail Stop 194-51 Dallas, TX 75265
1	Martin Marietta Aerospace ATTN: Mr. Dan Dorfman P.O. Box 5837 MP 113 Orlando, FL 32855
3	Matra Datavision ATTN: S. Grief R. McPherson M. Suarez 99 South Bedford Street Burlington, MS 01803
3	Mathematical Applications Group, Inc (MAGI) ATTN: M. Cohen R. Goldstein H. Steinberg 3 Westchester Plaza Elmsford, NY 10523
1	Megatek Corporation United Telecom Computer Group ATTN: S. Bryant 888 Washington Street Dedhan, MA 02026
1	Megatek Corporation United Telecom Computer Group ATTN: M. Landguth 3985 Sorrento Valley Blvd San Diego, CA 92121
1	Megatek Corporation United Telecom Computer Group ATTN: J. Phrohaska 7700 Leesburg Pike, Suite 106 Falls Church, VA 22043

DISTRIBUTION LIST

No. of Copies	Organization
1	Micro Electronics of North Carolina ATTN: Gershon Kedem P.O. Box 12889 Research Triangle Park, NC 07709
1	MIT ATTN: Dr. S. Benton RE15-416 Cambridge, MA 02139
1	Northrop Corporation Aircraft Division ATTN: Mr. Starr Mail Station 3501/84 1 Northrop Avenue Hawthorne, CA 90250
1	Northrop Corporation Research and Technology Center Manager, Autonomous Systems Laboratory ATTN: James R. Reis One Research Park Palos Verdes Peninsula, CA 90274
1	PDA Engineering ATTN: Lou Crain 1560 Brookhollow Drive Santa Ana, CA 92705
1	PRI, Inc. ATTN: W. Bushell Building E4435, Second Floor Edgewood Area-APG, MD 21010
1	Interactive Computer Graphics Center Rensselear Polytechnic Inst. ATTN: M. Wozny Troy, NY 12181
1	RGB Associates, Inc. ATTN: R. Barakat Box B Wayland, MA 01778
3	Structural Dynamics Research Corp (SDRC) ATTN: R. Ard W. McClelland J. Osborn 2000 Eastman Drive Milford, OH 45150
1	Sigma Research Incorporated ATTN: Dr. Richard. Bossi 8710 148 Avenue NE Redmond, WA 98052
1	Sikorsky Aircraft Division of United Technologies ATTN: R. Welge North Main Street Stratford, CT 06602
1	System Planning Corporation ATTN: A. Hafer 1500 Wilson Blvd Arlington, VA 22209
2	TRW Operations & Support Group ATTN: K. Dankers T. Heim One Space Park Redondo Beach, CA 90278
1	Vought Corporation ATTN: Paul T. Chan P.O. Box 225907 Dallas, TX 75265
1	Mr. John Bosma Cardinal Communications Corporation 12913 Eagle Dancer Trail NE Albuquerque, NM 87112
1	Professor Henry Fuchs University of North Carolina 208 New West Hall (035A) Chapel Hill, NC 27514

DISTRIBUTION LIST

No. of Copies — Organization

3 The University of Utah
Computer Science Department
ATTN: R. Riesenfeld
E. Cohen
L. Knapp
3160 Merrill Engineering Bldg
Salt Lake City, UT 84112

1 Science Applications, Inc.
ATTN: Terry Keller
Suite 200
1010 Woodman Drive
Dayton, OH 45432

1 Science Applications
International Corp
ATTN: Dr. Robert Turner
Suite 200
1010 Woodman Drive
Dayton, OH 45432

1 TASC
ATTN: Eric Keydel
1 Jacob Way
Reading, MA 01867

1 Thomas Hafer
1500 Wilson Blvd.
14th Floor
Arlington, VA 22209

1 XONTECH
ATTN: John Dagostino
1701 N. Fort Myer Drive
Suite 703
Arlington, VA 22209

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-C, A. Reid
AMXSY-CS, P. Beavers
AMXSY-D
AMSXY-G, W. Brooks
J. Kramar
L. Kravitz
AMSXY-J, H. Lee
AMXSY-RA, R. Scungio
M. Smith