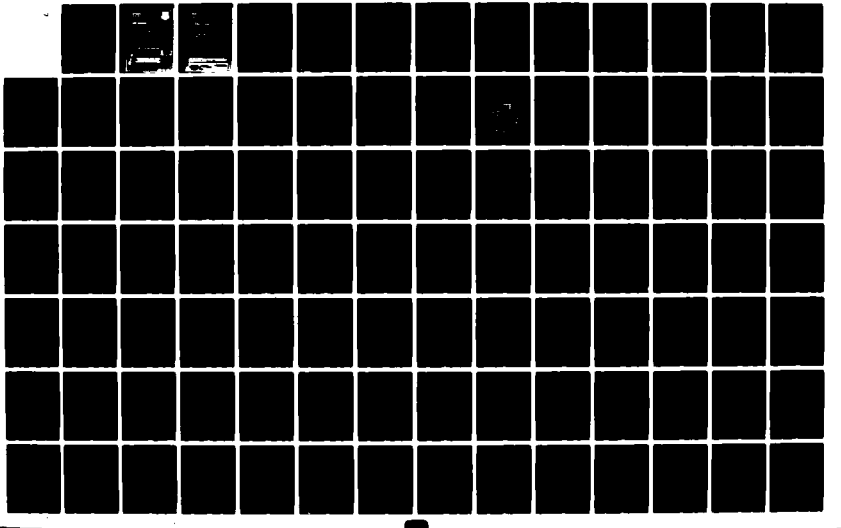
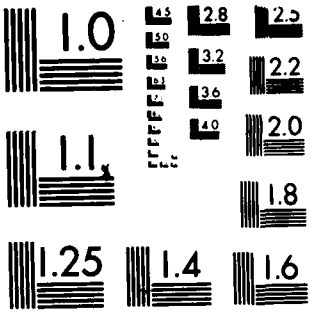
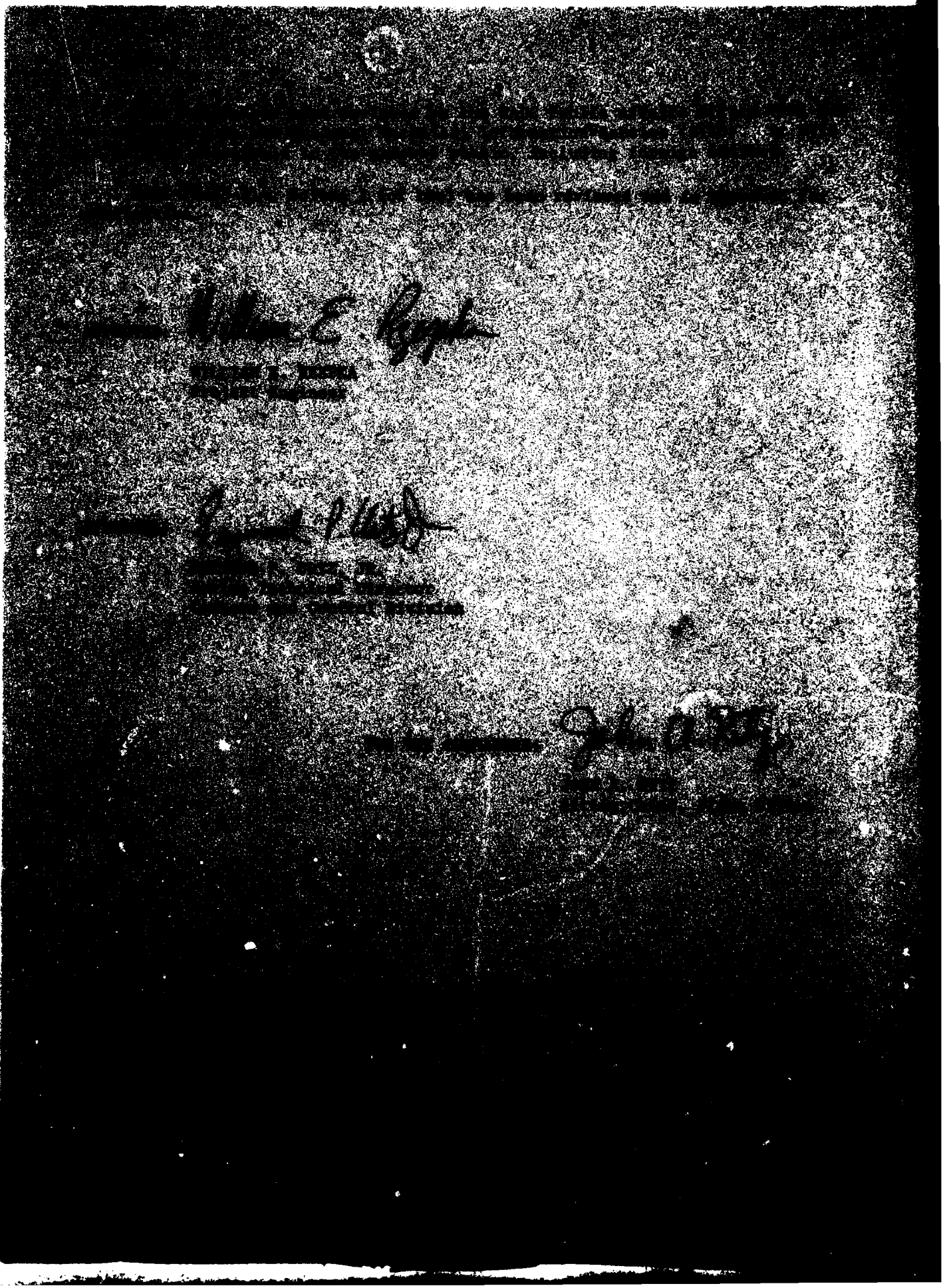
AD-A141 631 SREM (SOFTWARE REQUIREMENTS ENGINEERING METHODOLOGY) 1/2
EVALUATION VOLUME 1(U) MARTIN MARIETTA DENVER AEROSPACE
CO A STONE ET AL. FEB 84 MCR-83-553-VOL-1
UNCLASSIFIED RADC-TR-83-314-VOL-1 F30602-80-C-0272 F/G 9/2 NL







UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-314, Vol I (of two)	2. GOVT ACCESSION NO. AD-A141 631	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SREM EVALUATION		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 Oct 80 - 15 Nov 83
		6. PERFORMING ORG. REPORT NUMBER MCR-83-553
7. AUTHOR(s) A. Stone D. Hartschuh B. Castor		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0272
9. PERFORMING ORGANIZATION NAME AND ADDRESS Martin Marietta Denver Aerospace P.O. Box 179 Denver CO 80201		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55812203
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE February 1984
		13. NUMBER OF PAGES 104
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: William E. Rzepka (COEE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Requirements Engineering
Specification Languages
Software Tools

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

In the fall of 1980, the Rome Air Development Center contracted Martin Marietta Denver Aerospace to perform a comprehensive evaluation of the Software Requirements Engineering Methodology (SREM). The objectives of the evaluation were to assess the capabilities of SREM for specifying the software requirements of large, embedded computer systems and to recommend improvements which would enhance its effectiveness. Specific evaluation criteria were developed to judge the effectiveness of the methodology,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

support tools and user training.

The approach included attending a SREM training course and using SREM to specify the software requirements for two Air Force systems. One of these was a large distributed computer system whose 515 page, English language system description resulted in 35,550 lines of formal requirements specification with over 3000 data base elements. The relatively small number of errors uncovered indicates the effectiveness of disciplined requirements analysis techniques and the capabilities of SREM for exposing subtle problems.

In general, it was found that the SREM was an effective vehicle for specifying and analyzing the software requirements of large embedded computer systems, especially descriptions of real world objects, data requirements and message processing. However, deficiencies were noted in the specification language, particularly in describing parallel and distributed processing requirements, in the "friendliness" of the user interfaces to the analysis (consistency/completeness) and simulation tools, in the performance of these tools and in the effectiveness of the training. Appropriate improvements to all of the functional deficiencies are recommended.

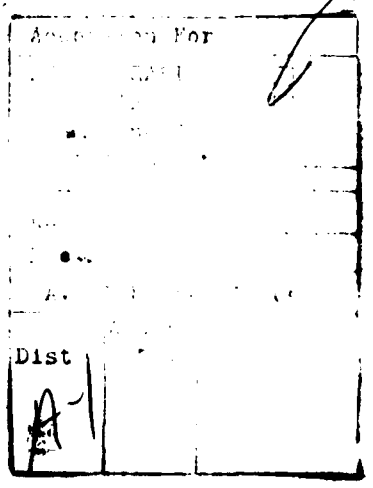


TABLE OF CONTENTS

1.0 INTRODUCTION

This document is the Final Technical Report of the Software Requirements Engineering Methodology (SREM) Evaluation project. SREM was evaluated for its appropriateness to Air Force applications in the command and control arena.

The SREM is a set of manual procedures and automated software tools that provides the following features:

- o The generation and independent validation of software requirements
- o The reduction of ambiguity and errors in software requirements
- o The ability to manage the requirements development process
- o Techniques to automate the process of validating software requirements

The first SREM version was produced by TRW, Inc. for the U.S. Army Ballistic Missile Defense Advanced Technology Center (BMDATC) in 1977. The system has been enhanced and improved since that time through successive versions. The 1980 SREM version was evaluated in this contract.

All work in the SREM evaluation project was performed by Martin Marietta Denver Aerospace under contract to the Rome Air Development Center (RADC). Funding for the evaluation effort was provided by RADC through Contract F30602-80-C-0272. This report is CDRL Item A002 of that contract. Technical direction and guidance for the contract was provided by Mr. William Rzepka and Mr. Roger Weber, both of RADC/COEE. The period of performance was October 1980 through November 1983.

Two objectives of the SREM evaluation effort were established by the Air Force. The first was to evaluate the use of SREM for describing embedded computer systems. The second was to recommend improvements to allow it to be a more effective system engineering tool.

To accomplish these evaluation objectives, three distinct aspects of SREM were examined:

- The ability of SREM to describe the properties of Command, Control, Communication and Intelligence (C^3I) software systems (detailed in Section 3.0);
- The most appropriate role for SREM in the Air Force software development life cycle (Section 4.0);
- The quality of SREM training currently available (Section 5.0).

Section 2.0 of this report summarizes the results, conclusions and recommendations of the evaluation project. The remaining sections (3.0 through 5.0) address each of the evaluation aspects in greater detail. A large set of appendices are bound in a separate volume. Appendices A through E contain an example of SREM application to an Air Force problem, the Narrow Band Emitter Locator Sensor (NELS) subsystem of the Advanced Sensor Exploitation (ASE) system. Appendix F contains observations on installing the VAX/VMS version of SREM. Appendix G describes the objective measures used to evaluate SREM-based designs and design specification alternatives produced during the evaluation project.

2.0 MANAGEMENT SUMMARY

2.1 Introduction.

SREM is a formal, integrated approach to requirements engineering activities. It begins when the system requirements analysis has identified system functions, the functional interfaces between subsystems, the system operating rules (conditional statements impacting when and in what sequence the functions are performed), and the top-level system requirements allocated to the computing resource. SREM is designed to induce certain qualities often lacking in the specification of many large software systems. The most important of these are:

1) Internal consistency, which provides agreement among descriptive statements in level and kind;
2) Explicitness, which requires unambiguous, complete descriptions of what is to be done, when, and with what kind of data;
3) Testability, which ensures that performance requirements are directly testable;
4) Traceability, which allows impact assessment of changes to system requirements.

In addition to the step-by-step requirements engineering techniques, SREM includes a machine-processable "English-like" Requirements Statement Language (RSL) and a Requirements Engineering and Validation System (REVS) to automatically process the requirements statements, and to perform a wide range of analyses and simulations on its centralized data base. REVS constitutes the automated tools part of SREM. It uses a relational data base (called the Abstract System Semantic Model or ASSM) to capture requirements processed from RSL.

Functionally, REVS consists of six software components that create the ASSM and examine its contents. These are identified in Table 2-1.

Table 2.1 SREM Functions

Mnemonic	Name	Purpose
RSL	RSL Translator	Translate requirements and create the ASSM
RSLXTND	RSL Extend	Augment/extend the RSL language elements
RADX	Analysis & Data Extraction	Extract info from ASSM for analysis/documentation; Identify ASSM subsets for consistency/completeness; Analyze ASSM content for data flow
SIMGEN	Simulation Generation	Build a simulation package
SIMXQT	Simulation Execute	Perform the simulation exercise
SIMDA	Simulation Data Analysis	Analyze and document the simulation

The primary mechanism for evaluating SREM consisted of applying it to two typical Air Force systems through the existing specifications that defined these systems. With this exercise, three evaluation aspects could be examined:

1) The ability of SREM to describe the properties of C^3I software systems (Section 2.2);
2) Where SREM can best be applied in the Air Force software development life cycle (Section 2.3);
3) The quality of SREM training currently available (Section 2.4).

Each of these aspects is discussed in the same format--goals of the assessment, methods used to perform the evaluation, and derived results and recommendations.

2.2 Applying SREM to C^3I Systems

2.2.1 Goals

To determine SREM's applicability to C^3I systems, two areas were examined--specification scope and requirements analysis capabilities. The first addressed the abil of SREM to describe all characteristics of C^3I systems. Th characteristics of C^3I systems that distinguish the rom other systems and the ones that the evaluation project tr express in RSL are:

1) RT (real-time) and NRT (near-real-time) processing;

2) The ability of the system to present data to an analyst and react to his decisions;

3) The ability to deal with real-world objects that simultaneously are users of C^3I data, providers of C^3I data and objects about which the C^3I system maintains data;

4) The ability to manipulate static data, such as cartographic or hypsographic information, as well as such dynamic data as tank position or sensor status;

5) Distributed processing;

6) Communications processing.

The second area of concern was SREM's capacity to automatically detect such specification errors as requirement ambiguities, requirement inconsistencies and specification incompleteness.

data ranges, are completed. RSL is again used to describe these requirements, referred to as Requirement Networks (R_NETs), into REVS which in turn updates the ASSM. The static analyzer (RADX) is executed to test for errors in element attributes or relationships between elements. When all information has been input and all errors corrected, the result is a formal functional requirements specification.

Before the requirements specification can be considered fully complete, a system simulation is needed. To perform the simulation, simple functional models are created in Pascal for each of the processing steps that have been identified. These models are embedded directly in the RSL and used as input to a simulation generation function to create a behavioral model of the functional requirements. The simulator is executed to verify that the envisioned system's interfaces and processing relationships behave as required.

In addition to verification of the processing relationships, performance requirements must be specified for the functional requirements so that system performance constraints can be tested. Each performance requirement will constrain a processing path within the system. The establishment of performance requirements by this method assures the traceability of requirements and highlights the system structure for review.

It may also be necessary to verify that the requirements specification is analytically feasible. To do this, one more simulation step, called the analytical feasibility demonstration, is performed. This simulation uses analytic algorithms of the intended system instead of simple functional models. The main goal is to establish that the identified software requirements can be tested.

All REVS components were used in evaluating SREM. The RSL translator function was used to construct an ASSM for each application, ASE and CSID. The RADX function served three purposes. First, the RADX analysis capabilities were employed to verify the consistency of the ASSM by examining the attributes associated with the ASSM elements. Errors detected could be traced to one of two places--the RSL description or the originating requirements documentation. Second, the same analysis capabilities determined the completeness of the formal specification by examining the relationships between elements. Third, the RADX capabilities for user-defined analysis were used to produce a document resembling a MIL-STD 490 Type B-5 specification. The flexibility of data requirements descriptions was enhanced by using the language extension function (RSLXTND) of REVS. The SIMGEN function was used in conjunction with the SIMXQT and SIMDA functions to exercise the data flow logic. Models of the RSL processing blocks were added to allow the SIMGEN function to construct a simulation program executed by SIMXQT with the results analyzed by SIMDA.

2.2.3 Results

For each system studied, the size of the specification used, the time spent creating the ASSM data bases and the size of the ASSMs produced are shown in Table 2-2.

Table 2-2. RSL Production Data

Spec	Document Size (pages)	Elapsed Man-Hours[1]	Time (Mths)	ASSM Size Lines RSL	Number Elements[2]	Percent Data[3]	Percent Proc[4]	Percent Other[5]
CSID	196	7498	10	10353	1109	53	40	7
ASE	515	11277	20	35550	3016	46[6]	8.8[6]	44.9[6]
Total	711	18775	30	45903	4125	--	--	--

Notes:

1 Engineering hours to translate and analyze specification

2 RSL elements such as DATA, R_NET.

3 MESSAGEs, DATA, ENTITY_CLASSes, etc.

4 R_NETs, SUBNETs, ALPHAs, etc.

5 SOURCEs, ORIGINATING_REQUIREMENTs, etc.

6 Averaged over six subsystems.

The following statistics can be derived from the table:

1) Approximately 19 lines of correct RSL were produced per eight-hour workday;
2) Approximately 1.8 correct RSL element definitions were produced per eight-hour workday;
3) While error occurrence data were not kept for CSID, about 100 specification errors were found when applying the SREM (not necessarily REVS) to ASE, resulting in about 120 man-hours expended per error discovered.
4) A significant disparity occurs in the composition of the data bases; in particular the amount of information required to express process descriptions (40% vs 9%) and overhead activities (7% vs 45%) varied significantly between specifications examined.

The learning curve for RSL semantics must be considered as well. While RSL was produced at approximately 11 lines/day initially, experienced analysts were able to achieve about 25 lines/day. A similar difference was experienced in the production rate of RSL element definitions (1.2 vs 2.1 elements/day).

Analyst effort to achieve these productivity rates averaged 26.4 man-hours per page of specification. Since the project consisted of only two data points, this is not considered a reliable predictor of validation effort.

The conclusion that can be reached from Table 2-2 is that the application of a disciplined methodology like SREM is a cost-effective means of analyzing software requirements. In comparison, Figure 2-1 shows a consolidation of error correction cost data for the Safeguard and Titan New Guidance missile systems. Cost to correct an error found during requirements analysis fell in the $100-200 range while correction of those found during final test and integration were on the order of $10,000 due to higher complexity and more lines of code affected. In the SREM test, expending 120 man-hours to correct an error early is clearly cost-effective.

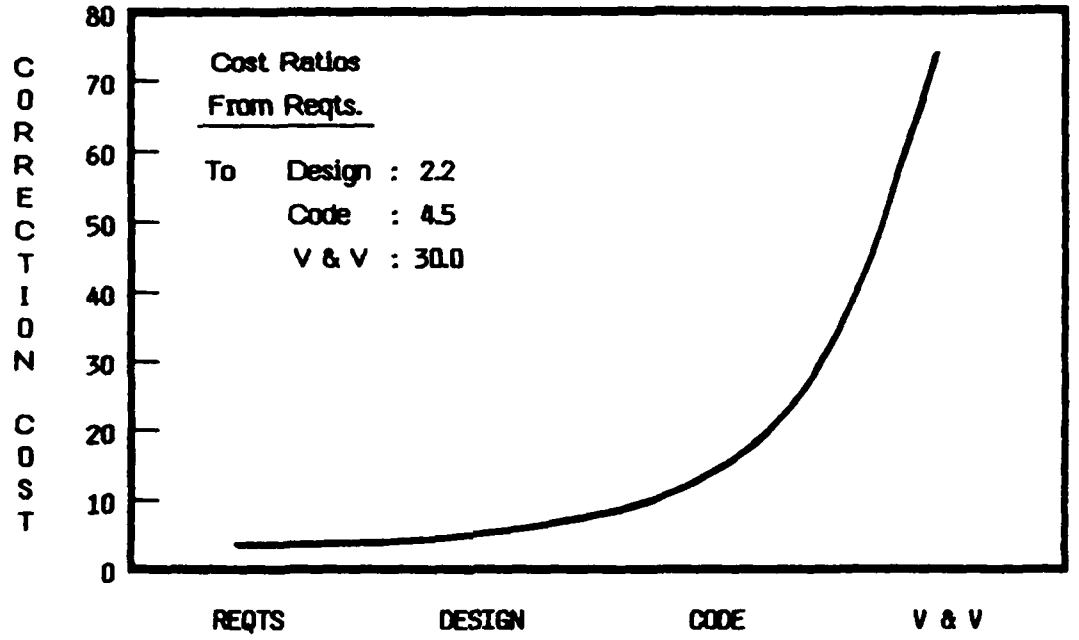


Figure 2-1. Error Correction Cost

A detailed analysis of design flaw exposure was done for the ASE system. Error classes and how they were found are shown in Table 2-3. Clearly evident here is the strength of SREM as a methodology as opposed to its automated validation tools. This should not be too surprising considering its early life-cycle application where expressive vigor is achieved only after extensive analyst design iteration to achieve comprehension.

Table 2-3. ASE Error Data Summary

Class	Number	% of Total	How Exposed
Inconsistency/Misuse	30	36	80% by analyst; 20% by REVS
Ambiguity/Lack of Detail	17	21	100% by analyst
Incompleteness/Non-use	35	43	100% by analyst
	82	100	

All of the attributes that characterize C^3I systems were contained in one or both of the specifications. The real-time and near-real-time phenomena present throughout the ASE specification proved very difficult to express because of the limitation on the description and operation of timing paths to R NET boundaries. Decision-making characteristics proved difficult to express when applied to man-machine interfacing in prompt-driven systems. Throughout translation of the CSID specification, the project team found it necessary to add state flags and their related conditional data to describe and correctly associate the CSID user's reply to the prompt that evoked the reply. The additions severely reduced the clarity of the requirements in the ASSM. C^3I real-world objects were easily transformed into RSL by using the element types ENTITY CLASS and ENTITY_TYPE. The data requirement characteristics were described easily by using attributes associated with the element types DATA and FILE. RSL is only capable of describing distributed processing and parallel process thread requirements in which the sequence of processing is not important. This generality in processing structure precludes detailed description of distributed and parallel processing situations where time sequences are important, and so restricts the appropriateness of SREM to only software requirements analysis or high level design activities. Communications characteristics are easily described

by using appropriate RSL attributes of specific element types; detailed information (e.g., baud rates) can be associated with the characteristics by appropriate extensions to RSL using the language extension function (RSLXTND).

2.2.4 Recommendations

The sample application of SREM to these test systems clearly demonstrated the benefit of disciplined techniques in requirements analysis. A formalized scheme uncovers many problems and questions that help improve the integrity of the system description. But this is indicative of any disciplined approach to requirements analysis; benefits outweigh costs because flaws are detected early and are not permitted to permeate the system.

The major recommendations that relate specifically to C^3I applicability are concerned with parallel processing and distributed processing capabilities. For appropriate detail, the RSL must be modified to allow explicit description of parallel and distributed processing. The RSL modifications will in turn require the modification of RADX to extract and analyze the new information, and the modification of SIMGEN so parallel and distributed processing can be portrayed within the simulations.

2.3 Role of SREM in the Software Life Cycle

2.3.1 Goals

The goal here was to determine in which phase or phases of the software life cycle SREM is most appropriate. This goal was addressed using three objectives:

1) Assessment of the methodological technique prescribed for using SREM;

2) Evaluation of the ability of the REVS software tools to support SREM;

3) Determination of the best placement of SREM within the software life cycle as used by the Air Force.

The term "software life cycle" is used to identify the processes that occur in the development of software systems from initial conception to final realization in an operational environment. The following phases and associated processes are considered:

1) Conceptual - The need for a system to solve a particular set of problems is identified, with feasibility assessments, tradeoff studies and analyses being performed. Requirements for computer resources are allocated. The description of the system takes the form of an initial system specification;

2) Requirements Definition - Requirements are defined for interfacing, performance, safety, human factors and others. The functions should be defined, and a data dictionary produced;

3) Design - The specification for the system envisioned in the requirements phase is transformed into an overall design of how the system accomplishes its goals. The transformation involves the allocation of system functions to hardware and/or software, a description of the objects the system is to operate on, and a description of the algorithms to be used in operating on those objects;

4) Coding and Checkout - The design is translated into a computer language. It is then executed, in single or combined elements, to evaluate its performance;

5) Test and Integration - The resulting program is tested to

ensure that the software performs as intended and the system, as implemented, fulfills all system requirements;

6) Operational - The system is in operation and must be maintained. The maintenance process is invoked to correct problems not previously encountered or to change the system as the needs of its users change.

2.3.2 Methods

As its name suggests, SREM is intended specifically for requirements analysis. However, requirements analysis involves functional specification and data description which occur in both the Requirements Definition and Design phases of the life cycle. The first part of evaluating the methodology addressed whether SREM was an effective technique for defining unambiguous and testable requirements. The second evaluation aspect was a comparative one, determining whether the requirements produced were of sufficient quality that derived designs would be of higher quality than those produced without the benefit of SREM. Hence, the use of SREM was evaluated by (1) examining the consistency and completeness of the requirements produced, (2) examining the REVS tool products for information content, tool maturity and utility, and (3) comparing designs produced from both the original specifications (CSID and ASE) and from the translations of those specifications into RSL.

To correctly place SREM in the Requirements Definition and Design phases of the software life cycle and to identify how well it supported requirements analysis, the completed requirements data bases (ASSMs) for the ASE and CSID were used as the basis for validating the original specifications by validating the corresponding RSL statements. Subsequently, the RSL was used to create a top-down NASSI-Schneiderman expression of the design.

Simultaneously, a design was produced in a conventional fashion using top-down decomposition expressed via Nassi-Schneiderman charts by using only the software specification documents. Figure 2-2 shows this comparison scheme. The resulting designs were compared on common bases using McCabe's complexity and Myer's reliability measures (Vol II, Appendix G), and by subjectively determining from design walkthroughs if the design characteristics of hierarchical construction and levels of abstraction were adequately supported by SREM.

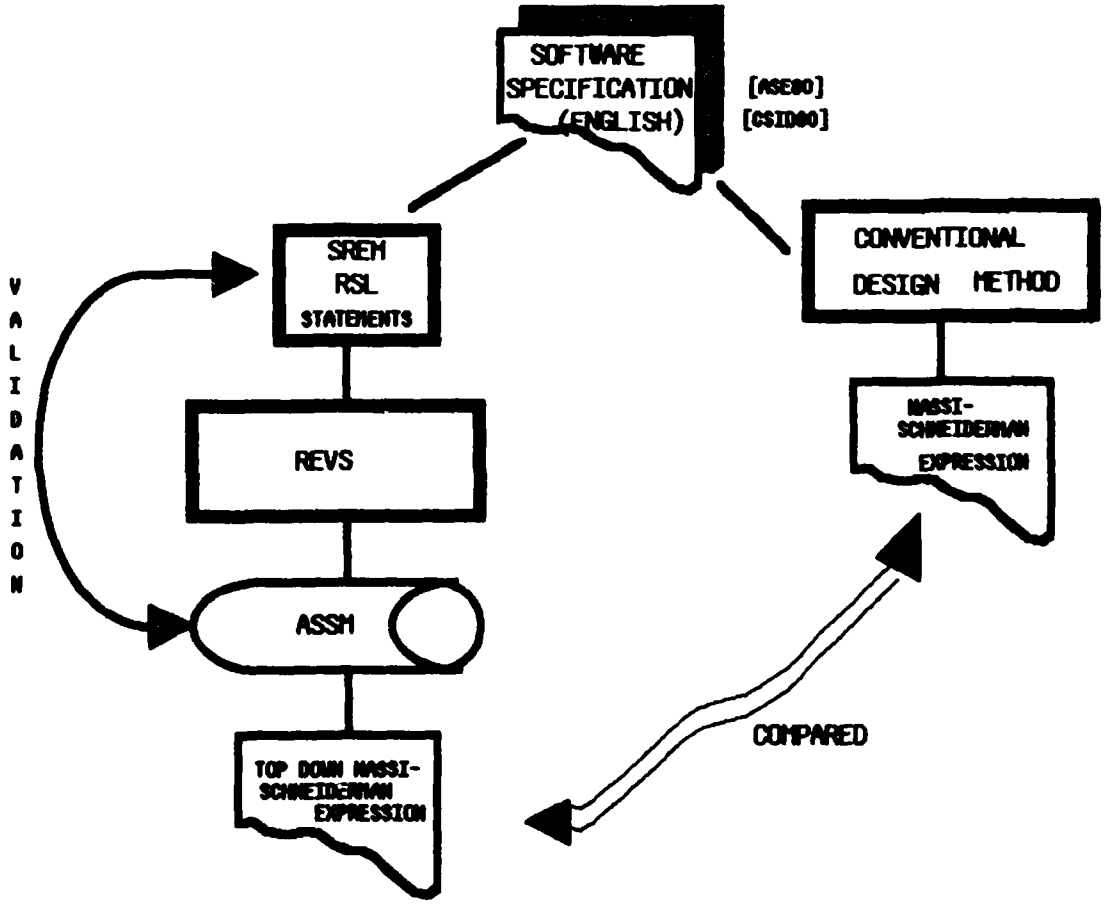


Figure 2-2. System Comparison Scheme

2.3.3 Results

The SREM proved to be of great value in the task of requirements analysis. This structured approach aided the requirements analyst in identifying inconsistencies and ambiguities in the specification as the methodology progressed through all of its steps. However, SREM utility appears to lie within a fairly narrow range of the life cycle. SREM was developed to specify software requirements after the system requirements analysis phase has been completed but before any detailed processing algorithms have been formed. The consensus of the analysts using SREM was that it is best used when defining, correcting or analyzing software specific requirements of the system.

The RSL Translator cannot adequately recover from a syntax error and continue its error detection process as if the previous error did not occur. As a result, the phenomena of RSL syntax errors being produced as a result of a previous error was prevalent throughout the entire specification translation process. For example, the omission of quote marks in a DESCRIPTION field causes subsequent lines to be flagged in error. This ripple effect caused the project team to become very sensitive to the quality of the information being entered into the ASSM and resulted in the RSL being evaluated for errors by the team and not by the automated language translator. Even though this meant that team members were doing syntax checking by hand, it is not the RSL syntax analyzer that is weak but the error detection component that needs improvement. The language translator for RSL has not reached a level of maturity such that it can discover syntax errors, recover from them and then continue with the translation.

The SIMGEN function of REVS was applied to two of the ASE subsystems, C^3I and NELS (Vol II, Appendix E). A relatively

complex chain of events is necessary to create a simulator (Figure 2-3). In addition to a validated ASSM, a Simulator Definition File (SDF) is needed. This, along with BETA and GAMMA definitions in Pascal are all input to SIMGEN to build the simulation. SIMGEN itself generates additional Pascal code to complete the definition. All of these then get passed to the simulator execution function (SIMXQT).

The SIMGEN function will automatically run an ASSM data analysis (RADX) every time a simulator generation is attempted. Valuable time and resources may be unnecessarily expended if no significant changes were made to the simulator definition (GAMMAs). Moreover, SIMGEN does not maintain any kind of "error status" flag on the results of the RADX analysis so that the simulator generation will be run to completion even though the preceding RADX step discovered an error.

In evaluating the designs produced from both RSL specifications and the original specification documents (Figure 2-2), no clear advantage of either technique arose (see Appendix G and Section 4.0 for more detail). For large systems of course the availability of a machine-readable data base of design information just for update purposes is beneficial. Both sets of designs were approximately equal in complexity and modularity according to accepted measures. However, a controlled experiment was not intended and the same design team was used for both versions. Influences certainly occurred as a result of this commonality.

2.3.4 Recommendations

While the SREM technique itself proved to be valuable for identifying specification problems, REVS left much to be

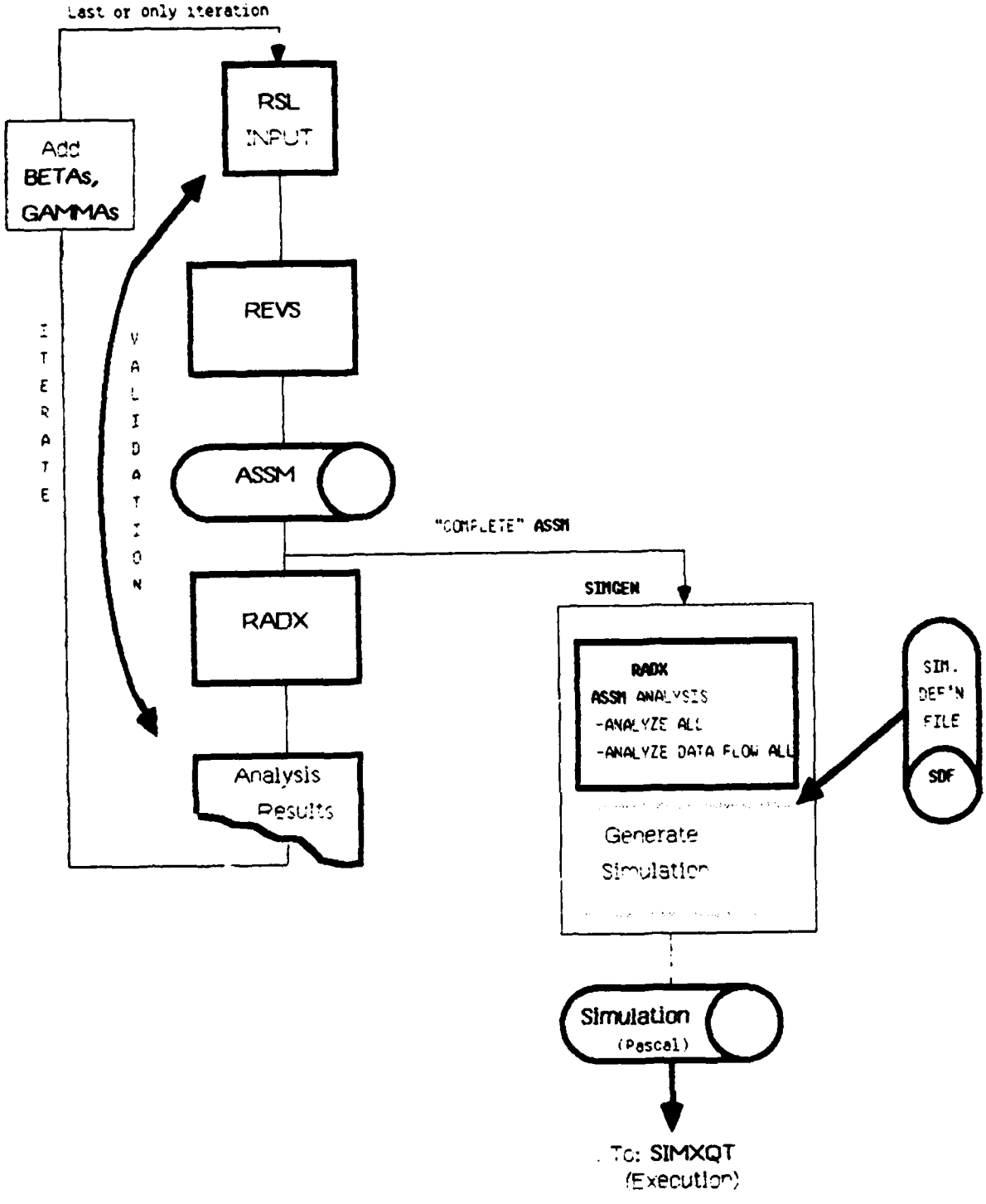


Figure 2-3 Simulator Generation Operation

desired. The RSL translator was inefficient and performed error recovery inconsistently; both of these problems should be corrected before REVS is released for general use. RADX faces similar problems. Data flow analysis should be decoupled from the SIMGEN functions and placed under user control. When a simulation's data flow has been analyzed once, and as long as it is not modified, the simulation should be executable without another data flow analysis.

In general, REVS needs to be made more efficient in terms of the computer resources used and its user-friendliness. Several basic structural changes would improve the REVS package. The changes to consider are:

1) Partition the predefined RADX analytics into smaller executable pieces so the VAX version is comparable in modularity to the CDC version;
2) Further investigate VAX multi-tasking to accomplish more parallel activities (expand on simple foreground-background concept);
3) The DBMS technique should not utilize its own paging scheme which encourages thrashing in a virtual memory system;
4) REVS initialization requires inordinate time indicating extensive processing each time it is started. Table driven or other predefined structures could reduce the resource consumption here;
5) The command language facilities would be improved with a hierarchical HELP structure, more detailed prompting levels (indicating which subsystems, providing prompt message of what is needed, signaling when function is complete), and elaborating numeric error codes with meaningful messages.
6) Decoupling RSL processing from data base creation (syntax only check) would reduce the time and costs associated with RSL translation.

Notice that some of these recommendations may be obviated by future implementation strategies.

2.4 Quality of Available SREM Training

2.4.1 Goals

Assessment of the SREM training course covered three main areas of concern. First, the planning and logical structure of the course was examined with respect to:

1) Audience considerations - What was the target audience and did the course address its needs? Was the amount of material covered reasonable in the time allowed?
2) Course overview - Were the objectives of both SREM and the training made clear?
3) Planned activities - Did the class exercises support the material covered in lecture? Was sufficient hands-on experience included?
4) Progress evaluation - Were criteria given so the students could evaluate whether they had learned the material? Were the criteria related to observable student behavior?
5) Course summary - Was provision made for summarizing or reviewing difficult concepts?

Second, course execution was considered, including:

1) Teaching methods - Was the quality and amount of instructor-student interaction adequate? Were discussions encouraged? Were examples used to clarify difficult concepts?
2) Environment - Were the conditions under which the course was taught comfortable and conducive to learning?

Third, course materials were evaluated for utility in both learning and using SREM.

2.4.2 Methods

The evaluation team attended the SREM training course to learn the methodology (with its associated software tools) and to evaluate the effectiveness of the course. The course was examined with respect to its own stated objectives of communicating software requirements engineering technology so students understand how to correctly define requirements and have the knowledge to use SREM and REVS correctly and comfortably. The course structure, presentation and materials were also evaluated. After the first two weeks, the design team "students" were polled for their reactions. Several "audit" attendees also submitted individual evaluations of the course. These included experts in requirements engineering technology, professional educators and others who were interested in the methodology. The observations of all attendees were incorporated in the final evaluation.

2.4.3 Results

Many of those attending the training course as it was given were interested in the methodology but did not intend to use SREM. Therefore, to cover the complete methodology, all of the scheduled initial two-week lectures were presented during the first week. Many black on white viewgraphs were used as the main instructional aid. Most of the viewgraphs were reproductions of selected pages from the SREM Management Overview [PASI80], which is essentially a user's course textbook and not an "overview" as the name implies.

Examination of the SREM training course with regard to the course structure, materials and execution uncovered a number of serious flaws. The structure and objectives of the course were not clearly presented to the students. The format of the course followed the general sequence of the SREM life cycle but it was not explicitly divided into planned lessons with objectives, learning activities and achievement criteria specifically geared to each lesson. Very minimal evaluation of learning was included in the course structure. Retention of the material was hindered because no definite framework was presented in which the students could place the concepts being taught. The course materials were too voluminous to be easily dealt with and were not organized or cross-referenced to enable the students to readily determine where information was located. The fact that course mechanics were based on viewgraphs, presented in a darkened room, contributed not only to eye fatigue but to overall poor retention.

2.4.4 Recommendations

The major recommendation concerning SREM training is that a complete restructuring of the training course on a criterion-referenced basis be made [MAGE62, MAGE67]. In lieu of a complete redesign, improvements can be made to render the existing course more effective. A course overview should be provided on the first day, including a description of course objectives in terms of learned student behavior, a course schedule and a subject matter overview including such topics as where SREM fits in the software life cycle, its underlying philosophy, the history of its development and the overall structure of SREM. This could also serve as a basis for a short high-level course for managers, customers and others who need not become skilled in the actual application of SREM. The material in [TRW79] would serve as an excellent basis for this presentation.

The sessions should be clearly divided into short lessons, each including its own overview, objectives, examples, learning activities and achievement criteria. Unfamiliar terminology (such as ENTITY_CLASS) and terms used in an unusual way (e.g., FILE) should be particularly emphasized so the students understand what is being done. Examples that clarify the meaning of each new construct should be liberally used in the lesson and the learning activities should include the application of new material to a single system example which is used throughout the course for continuity. The lessons' place in the methodology must also be clearly shown by sequencing them to follow the phases of SREM application.

Students should receive the following materials for the training course, fully cross-referenced for use during and after the class:

1) A course guide showing a schedule and a course overview;
2) A text, in lesson sequence, including techniques, constructs and examples of SREM application;
3) A description of an appropriate example software system to be used throughout the course for application exercises;
4) Enough forms to complete all application exercises;
5) Reference materials describing syntax, semantics, processing and error messages;
6) A glossary, with examples, of SREM terminology.

If viewgraphs are used as lecture aids, the number of transparencies and the density of information on each should be greatly reduced. This will minimize the amount of time students must spend in the dark and make a higher level of teacher-student interaction possible.

3.0 APPLYING SREM TO C^3I SYSTEMS

3.1 Goals

SREM was applied to two system specifications, CSID and ASE. As the specifications were being translated into RSL and the resulting requirements data bases analyzed and compared, the analysts were evaluating the ability of SREM to describe and employ the following C^3I characteristics:

1) RT (real-time) and NRT (near real-time) - The need for systems to assimilate and distribute information within strict time constraints;

2) Decision-Making - Aspects of a system that deal with the man-machine interface and the ability of the system to present data to an analyst and react to his decisions;

3) Operational Entities - The ability of a system to deal with such objects in the real world as tanks and aircraft that are simultaneously users of C^3I data, providers of C^3I data and objects about which the C^3I system maintains data;

4) Data Requirements - The ability of the system to store and manipulate static data, such as cartographic or hypsographic information, as well as such dynamic data as tank position or sensor status;

5) Distributed Processing - Portions of the system that deal with the distribution of both processing and data over one or more nodes of a computer network;

6) Communications - The ability of a system to perform message handling, buffering or generation, and to deal with the idiosyncrasies of modern high-speed communications gear.

3.2 Methods

In the generation of RSL from the specification documents, ([ASE80] and [CSID80]), the methodology described by the Software Requirements Engineering Methodology (SREM) was employed to keep the formal specifications of the systems consistant with the source documents. All of the characteristics that make C^3I systems unique were contained in one or both of the specifications. The consistency between translations was important because all of the RSL, and hence, the resulting data bases, were considered as the specification source for subsequent designs.

The SREM process is illustrated in Figure 3-1. It starts with a system specification that can be a formal specification of a system, a conversation with the intended user, or a mental image of the system. The specification is translated and interpreted to determine the interfaces with the outside world, the messages across these interfaces and the required processing relationships and flows. (Phase 1). The formal language RSL is then used to describe these requirements to REVS, which translates the RSL and captures the requirements in a data base called the Abstract System Semantic Model or ASSM.

This initial activity produces a nucleus of information called the "kernel", which constitutes the minimum needed to document the major elements of a functional requirements definition. The RADX program is then used to evaluate this kernel (Phase 2). During this process, specific problems in the system specification will be found, e.g., ambiguities and inconsistencies. These problems are corrected by an iterative process until the specification of the system is

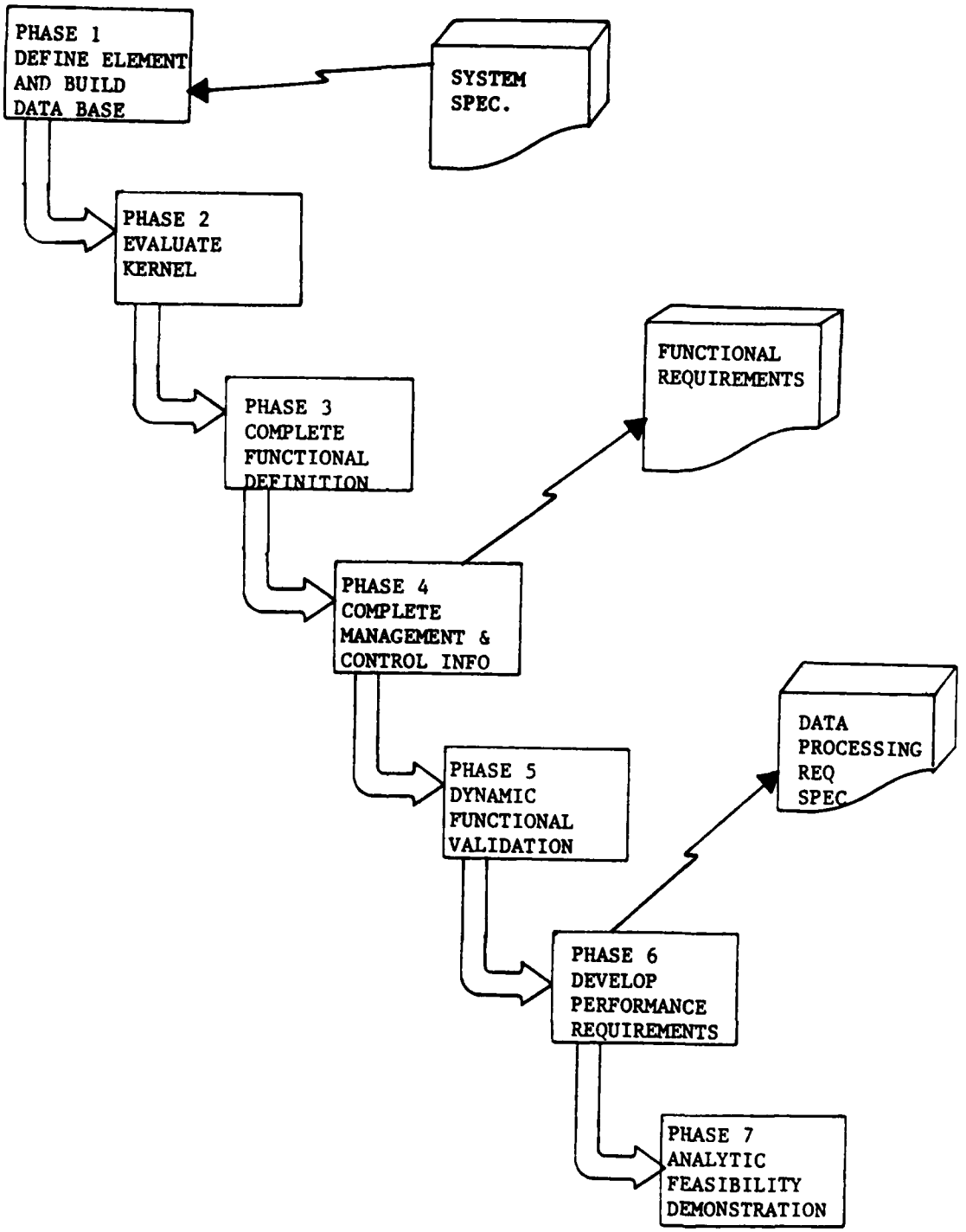


Figure 3-3. SREM Phases

satisfactory. Next, the details of the functional requirements, including all of the input/output data relationships, the processing steps, the attributes and maximum/minimum values, and the allowed data ranges, are completed. The static analyzer (RADX) is again executed to test for errors that exist in element attributes or relationships between elements (Phase 3). Phase 4 complements this completion process by specifically addressing the traceability of original requirements to RSL statements and decision rationale for requirements which may be excluded. When all information has been input and all errors corrected, the result is a functional requirements specification which is complete and consistent as defined by SREM.

The purpose of Phase 5 is to verify the completeness of the functional requirements by validating their dynamic properties. This is done by modeling the interaction of the data processor component with its environment by simulating the sequences of messages in and out of the processor. To perform the simulation, simple functional models (named BETAs) are developed in the Pascal programming language for each of the processing steps identified in Phases 3 and 4. These models are input to a simulation generation function to create a "simulator" of the functional requirements. The simulator is executed to verify that the envisioned system's interfaces and processing relationships perform as required. In addition to verification of a system's processing relationships (Phase 5), there is a need to establish constraints (Phase 6) on those functional requirements that specify performance criteria so that system performance constraints can be tested and traced. Each performance requirement will constrain a processing path within the system (named a VALIDATION_PATH) and have associated with it points (VALIDATION_POINT) from which information is extracted. These information points consist of those data elements identified in Phases 3 and 4 that are the pass or fail criteria for the constraint. The establishment of performance requirements by this method assures the traceability of requirements and high-lights the system structure for review.

It may also be necessary to verify that the formal SREM specification is analytically feasible. To do this, one more simulation step, called the analytical feasibility demonstration (Phase 7) is performed. This simulation will use analytic algorithms, named GAMMAs, of the intended system instead of functional models. It need not run in real-time, but should consist of real algorithms. The main goal of Phase 7 is to establish that the identified software requirements can produce a design which is testable and meets the functional and accuracy requirements of the system.

Following the methodology as described in the Management Overview [TRW79], definition of the ASE and CSID systems in RSL began by extracting originating requirements from the specification documents and recording the page and paragraph location as the source of the requirements information. The next step was the description of interfaces, the messages that cross them, and the information contained in each message. Then, items such as DATA or FILEs that constituted the messages were defined. Operational entities such as the cartographic data base in ASE were represented as ENTITY-CLASSes because the element ENTITY_CLASS associates FILE and DATA information into blocks of information that can be accessed by any process. The process of correlating data requirements with other RSL elements was, for the most part, straightforward. The data requirements translated well into RSL and usually became RSL DATA and FILE elements.

The expression detail in the two specifications were represented differently. The CSID system was described to the design level using a Program Design Language (PDL) while the ASE description was a functional decomposition using DeMarco diagrams. In the application of SREM, PDL procedure blocks become R_NET structures, as did many of the DeMarco processes. The DeMarco processes not broken down in a DeMarco diagram became ALPHAs. Likewise, single PDL statements involving lengthy or complex computations often become ALPHAs in order to capture the computation in a single process. In the DeMarco

diagrams, the elements describing communication between processes (ALPHAs) were most often identified as simple DATA or FILE elements. The more complex MESSAGE element was used if the receiving processing block did little more with the communication than pass it to an OUTPUT INTERFACE. The real-time and near-real-time characteristics were described in RSL by using VALIDATION_PATH and VALIDATION_POINT elements to identify segments within the R_NET structures that were time-critical.

3.3 Results

While the SREM was being applied to the ASE and CSID descriptions, several statistics were accumulated. Table 3-1 illustrates the size of the specifications, the effort expended in translating them into RSL (in man-hours), the elapsed time to translate them into RSL (in months) and the amount of RSL produced in terms of the number of lines input to the translator and the number of elements described (such as process blocks or data descriptions). Several conclusions can be reached from these data:

1) The average production rate of RSL statements (approximately 19.5 lines/8-hour day) can be compared to the range of 32-54 (source lines of programming language) in [WARB83], 8-32 lines (for jobs larger than 10K source lines) in [NELS78], or 12 lines per day reported in [TAUS80];
2) The learning curve for RSL semantics is quite severe. RSL is produced at the approximate rate of 11 lines/day when the analyst is learning it (see CSID data) but jumps dramatically to 25.2 lines/day once the analyst has become experienced (ASE data). Element/day rates confirm the line/day statistics -- CSID analysts (while learning) produced about 1.2 elements/day, the ASE (having learned) rate was 2.1 elements/day;
3) Software requirements analysis is a highly labor intensive

activity. Analysts salaries, overhead etc ($372,000) consumed approximately 95% of ASE system evaluation costs. The associated computer generation and analysis of the formal requirements specifications accounted for the remainder ($20,400).

The following observations were also noted concerning the specifications. The number of RSL elements/page of specification is remarkably consistent (5.6 for CSID, 5.8 for ASE) despite different authors and styles. Also, at present there is no reliable predictor of the effort required to validate specifications. In this evaluation about 26.4 man-hours/page of specification were expended, but the project represents only two data points so any conclusions are tenuous at best.

Table 3-1. RSL Production Statistics

System	Spec Pages	Effort	Elapsed Time	RSL Produced	
				Number of Lines	Number of Elements
CSID	196	7498	10	10353	1109
ASE	515	11277	20	35550	3016
Total	711	18775	30	45903	4125

A detailed examination of the RSL yielded the data in Table 3-2. This data confirmed the intuitive perception that, on average, approximately the same amount of effort was expended in data definition (47%) and overhead activities (39%), such as requirements traceability, configuration management and textual descriptions, each of which far exceeded the amount of effort devoted to the specification of control processing (13%). This relationship is particularly emphasized for the ASE, where averages over the six subsystems are 46%, 45% and 9% respectively.

Table 3-2. ASSM Composition

	Composition		
System/Subsystem	Percent Data	Percent Processing	Percent Other
CSID	53	40	6
ASE/ASET	61	14	25
/C3I	37	8	55
/MTI	49	9	42
/NELS	56	6	38
/T&C	41	13	46
/ES	34	3	63
Average	47	13	39

Table 3-3 presents some data concerning the errors uncovered during the validation of the ASE specification. Similar data was not collected for the CSID. The implication of the data is that approximately 137.5 man-hours (or slightly more than 3 man-weeks) were spent in discovering each error. This amounts to a cost of approximately $4785 to detect each error. This figure appears to have been influenced by several factors. First, many of the textual descriptions in the ASE system specification were generated from DeMarco data flow diagrams, resulting in a high quality specification with relatively few errors. Second, the analysts reported that not all of the errors detected (about 10%) were documented. Third, in arriving at the cost/error figure, it was assumed that all of the man-hours (and hence costs) associated with the ASE evaluation were devoted to error detection. However, a significant portion was expended on other activities, such as configuration management, traceability and textual descriptions. A final perspective from which to view this data is the total cost of developing the ASE system. To date, approximately $7,500,000 has been invested in the design and

implementation of ASE. Hence, the cost of using SREM to specify and analyze the ASE system has been about 5% of the total system development cost.

Table 3-3. ASE Error Data

	Errors Found		
Error Class	Number	Percent of Total	Method Used to Find
Inconsistency or Misuse	30	36	80% found by analyst applying SREM 20% REVS found
Ambiguities or Lack of Detail	17	21	100% found by analyst applying SREM
Incompleteness or Non-use	35	43	100% found by analyst applying SREM
Algorithm Flaws	N/A	N/A	ASE does not specify algorithms
Total	82	100	

As presently implemented, the ability of REVS to express and handle real-time and near real-time constraints is limited. The VALIDATION_POINTs which bracket a VALIDATION_PATH can be used to restrict the timing requirements to one or more processes (ALPHAs) within an R_NET or to the entire R_NET. Only in the case of an R_NET enabling another via an EVENT can a VALIDATION_PATH extend into more than one R_NET. But VALIDATION_PATHs cannot cross interfaces and so by definition are bounded by a single R_NET structure. Validation of the timing constraints of a complex C^3I system through separately identified networks severely reduces the information known about the interrelations of the system. However, it may be argued that representation of timing constraints is a rather detailed design activity, which would restrict the applicability of SREM to early design efforts.

The handling of decision-making, in regard to man-machine interfacing, is difficult using the existing stimulus-response constructs of SREM. This is because a decision choice is inherently data content sensitive, and because that choice can influence

subsequent processing paths. Since the human role must be modeled by an OUTPUT_INTERFACE element (a terminal RSL structure node), a response from that interface can only be returned through the "top" of the R_NET (INPUT_INTERFACE), and not directly back to the originating process for the message. The only way to tie the origin of decision choice (stimuli) information to response information is by textual comments which reflect the nature of the affect of the decision choices on the subsequent processing. Therefore, a requirements engineer must spend extra time carefully documenting all such sequences, structures, message processing priorities, etc. to ensure that the effect of decision choices is properly conveyed to consumers of the RSL specification information.

The collection, manipulation and dissemination of information about application-specific entities can be easily handled by use of the ENTITY_CLASS and ENTITY_TYPE elements. These elements are intended to deal with any information class whose status or contents can evolve or otherwise undergo state changes over time. The requirements encountered in the ASE and NELS subsystems to establish and manipulate data bases and queues were readily handled by equating such entities to ENTITY_CLASSes and ENTITY_TYPEs.

The description of data requirements in RSL is straightforward. Element attributes such as TYPE, RANGE, INITIAL_VALUE, UNITS, MINIMUM_VALUE, and MAXIMUM_VALUE greatly aid the detailing of data information. In addition the attributes of DESCRIPTION, PROBLEM and COMPLETENESS allow the requirements analysts to address capabilities and areas of concern to future users of the RSL information. The capability to extend the semantics of RSL via the RSLXTND function can greatly aid the user by allowing him to describe specific data requirements as newly introduced attributes and relationships peculiar to a particular project.

Requirements for parallel processing can be represented using the

AND construct to show mutually exclusive process sequences within an R_NET. The requirements for distributed processing can be represented in RSL by using multiple R_NETs that communicate via MESSAGEs. This representation allows for a very obvious separation in processing but the description of the processing sequence may not be clear if the specifications to coordinate the process threads do not exist. No existing RSL structure (beyond the DESCRIPTION attribute) is capable of explicitly relating or evaluating multiple cooperating process threads. To rely on textual description for such an important aspect is not adequate. Further, RSLXTND cannot be used to improve the semantics because it cannot affect the contents of RSL structures. Again, this further supports the applicability of SREM to only high level design.

RSL is very good at handling general details of message handling such as message origin, content and destination. Because there is no explicit link between a message output from one subsystem and a corresponding message that is input to another subsystem, the degree to which communication characteristics can be automatically traced is limited. The use of naming conventions to identify either a message origin or destination proved to be a very practical means to textually trace messages. For example, a message on troop movements produced by subsystem A and intended for subsystem B would be named TROOP MOVEMENTS_MSG_OUT. This name reveals basic information on the message content and processing direction. Automatically included in the creation of this message is the attribute CREATED BY that ties the message to the subsystem. The corresponding message to be received by subsystem B would be named TROOP_MOVEMENTS_MSG_IN. Again, the name reveals basic information on the message content and where it was created. The "TROOP_MOVEMENTS" parts identify the two messages as having identical information, and the MSG_IN, MSG_OUT parts identify the destination and the source, respectively. More discussion of naming conventions use appears in Section 4.3.5.

All of the C^3I characteristics were used during the specification translation into RSL and RSL does contain elements that can be used to describe most C^3I characteristics. The limitations of RSL in adequately describing C^3I characteristics lie in the realm of real-time, near-real-time, and the decision-making aspect of the man-machine interface. Translation of the ASE subsystems uncovered a serious deficiency in that the description of real-time and near-real-time characteristics is constrained to individual R_NETS. The translation of the CSID system revealed the problem of adequately matching the stimuli information to the response (decision) information.

3.4 Recommendations

As long as the basic constraints of the SREM modelling approach are understood and accepted, no specific recommendations can be made. In general, the scope of the process coordination and timing specification constructs could be increased to encompass more than one R_NET.

For increased clarity and human comprehension of the RSL, a more explicit description of parallelism is needed. By including a keyword IN_PARALLEL, the processing requirements would become more specific. For example, suppose processes A, B and C must be done in parallel and further that process B contains two processes B1 and B2, which can be performed in either order. This requirement would be specified as follows:

```
IN_PARALLEL
     [process A]
AND
     DO
          [process B2]
     AND
          [process B1]
     END
AND
     [process C]
END
```

SREM should be used to describe C^3I systems because the methodology is sound for such applications. The use of RSL for the description of C^3I characteristics is reasonable if it is understood that the detailed design characteristics of real-time, near-real-time and man-machine interfacing must be well described in English text to assure that requirements associated with these characteristics are understandable. The SREM approach is particularly well-suited to modelling multiple process threads as long as each thread consists of basic sequential processes or simple parallel activities.

SREM is particularly recommended as a disciplined technique for isolating specification errors. The cost early in the life cycle can result in significant savings later on.

4.0 ROLE OF SREM IN THE SOFTWARE LIFE CYCLE

4.1 Goals

The methodology as described in the SREM Management Overview [PASI80] was used to determine the effectiveness of SREM in identifying, describing and analyzing requirements. The collection of software tools, the Requirements Engineering and Validation System (REVS), was evaluated in terms of information content, maturity and usability. The Requirements Statement Language (RSL) was evaluated in terms of the RSL syntax, RSL structure and RSL concepts.

Another effort was to determine if SREM was capable of producing a MIL-STD 490 Type B-5 computer program development specification. The evaluation centered on the amount of information that could be extracted from an ASSM and utilized to produce a Type B5 specification.

4.2 Methods

4.2.1 Introduction

The Software Requirements Engineering Methodology was used to translate specifications into RSL, analyze the resulting ASSMs and generate a simulator from the ASSM. The ASSM resulting from a specification translation was considered as a formal definition of design. The original specification was also used as a design statement. The resulting designs were then compared to identify which method (the ASSM or the original specification) produced the design that best reflected the specification requirements.

The Software Requirements Engineering Methodology was used exactly as defined. No modification was made to the methodology.

4.2.2 ASSM Analysis

As the requirements for a software system are being identified and entered into the ASSM data base, the use of the REVS capabilities for automated ASSM analysis and simulation allows one to access and view the requirements as they are developed. The analyses and subsequent simulation of the system requirements are the objective "measures" used to validate the correctness and usefulness of the requirements. The following discussions deal with the building and analyzing of the ASSMs for the software requirements specifications ([ASE80] and [CSID80]), as well as the building of simulators for the C^3I and NELS subsystems of the ASE.

The ASSM analysis provided by the RADX function of REVS is divided into two types. The first type, called data analysis, checks the relationships and attributes of the elements contained within the ASSM. For example, a check is made to ensure that MESSAGEs contain DATA, that those MESSAGEs are properly formed, and that they are passed through an interface. The second type of ASSM analysis provided is data flow analysis. The data flow analysis traces DATA and FILE elements as they are processed by the elements (nodes) of an R_NET structure. All inconsistencies, such as not using an element, or the use of an element in an ALPHA that is not input to that ALPHA, are considered errors and indicated as such. Both types of ASSM analysis (data and data flow) were employed throughout the specification validation effort.

The RADX analyses were performed under two sets of circumstances. For the CSID specification validation, the CDC implementation of REVS was used. For the ASE, the VAX

implementation was used. These implementations differ in the way that pre-defined analyses are supported.

In the CDC implementation of SREM the automated data analysis has been divided into six phases. Each phase has been designed to determine the completeness and consistency of the attributes and relationships of several RSL element types. A summary of the tests that the phases perform which indicate the misuse or lack of ASSM elements follows:

1) Loop Detection - to check that the RSL elements used within an R_NET do not create looping conditions by i) direct/indirect referencing or ii) recursive definitions. An example of referencing is: SUBNET A calls SUBNET B, and within SUBNET B there is a call to SUBNET A. An example of recursive definition is:

```
DATA a              DATA d
     INCLUDES       INCLUDES
          DATA c              DATA b
          DATA d              DATA a
```

2) Use Designation - to insure that the RSL DATA elements identified (via the USE attribute) for a simulation (BETA, GAMMA) are used at that level of data detail and no lower. For example:

```
DATA POSITION
     INCLUDES
          DATA x
          DATA y
     USE BETA
```

shows that the data POSITION is the level of data detail to be

used in the BETA simulation and checks will be made to insure that DATA x and DATA y are not used.

3) Locality - checks that those DATA and FILE elements that use the LOCALITY attribute, and are associated with an ENTITY-CLASS, ENTITY-TYPE, MESSAGE, or FILE, have the proper scope value (global, local). Intrinsically, the locality of MESSAGE and FILE information is local (upon access of the element) and the locality of ENTITY- information is global (upon creation of the element). The locality test insures that any DATA or FILE element associated with an ENTITY-CLASS or ENTITY-TYPE has a global LOCALITY, and a LOCALITY of local if associated with MESSAGE or FILE.

4) Membership - insures that DATA elements are members of only one data set, i.e., DATA cannot be contained in more than one FILE, DATA cannot be associated with more than one ENTITY-CLASS, DATA cannot make a MESSAGE and be associated with an ENTITY. It is illegal for DATA elements to have multiple memberships. It is also important to note that messages are not considered as one data set, so a DATA element may be contained in more than one MESSAGE.

The VAX implementation of SREM has a single RADX consistency check that incorporates all of the previous tests. The single RADX consistency check (hence, an all or nothing condition) plus the knowledge of the large expenditure in time and cost of running this check, prompted us to employ the user defined set capabilities within RADX to obtain similar pertinent information. Table 4-1 identifies the sets used. The user-defined sets were created by analyzing and then subsetting the automated static analysis sets.

Table 4-1. User-Defined Sets for Data Extraction

SET GENERATED	INSURES THAT
INPUT_INTERFACEs that enable	all R_NETs are used
INPUT_INTERFACEs that pass	all INPUT_INTERFACEs are used
OUTPUT_INTERFACEs that pass	all OUTPUT_INTERFACEs are used
MESSAGEs without forms	all identified MESSAGESs have been made
MESSAGEs without made	all MESSAGEs must contain information
FILEs without contains	all FILEs must contain information
DECISIONs without choice	DECISIONs must be adequately defined and reasoned
ORIGINATING_REQUIREMENTs without traces	all identified requirements must be used
ORIGINATING_REQUIREMENTs without sources	all identified requirements link to the specification
ALPHAs without refers	all ALPHAs are used within a net
ALPHAs with inputs only	all ALPHAs are completely described
ALPHAs with outputs only	all ALPHAs are completely described

4.2.3 Simulation Techniques

Through the SIMGEN function SREM users have available two types of discrete event simulators. The two types, BETA and GAMMA, model different levels of system requirements. The BETA simulator, which requires user-supplied models written in Pascal, handles functional simulation of the processing described in the ALPHAs (RSL processing definitions). The GAMMA simulator, also written in Pascal, is used for analytic simulation. It requires models that employ algorithms that mimic those that will be used in the actual software system. It should be noted that the GAMMA simulator does not establish real-time feasibility of the set of algorithms to be used, but is used as "an existence proof of an analytic solution to the problem [ALF079, pg 9-68]."

All simulators generated by REVS, either BETA or GAMMA, are composed of five components: R_NET procedures, Simulation Executive, Simulation Event Manager, Simulation Data Manager and Simulation Driver. An R_NET procedure, which is represented by a BETA or GAMMA model, is the only RSL element executable in a simulator. The Simulation Event Manager schedules an R_NET's procedures for execution. The Simulation Data Manager controls all RSL DATA constructs. The Simulation Driver interfaces with the R_NET procedures, and the Simulation Executive is responsible for overall simulation control, including the simulation clock.

The preparation of an ASSM for a simulation generation is divided into two parts -- an automated ASSM check for element completeness and consistency and creation of the Simulation Driver Definition File (SDF). The checks for completeness and consistency accomplished by the automated RADX phase runs and data flow analyses will indicate ambiguities and errors that exist in the ASSM. The SDF is written externally to REVS in the Pascal language. The SDF is a counter part of the existing R_NETs in that it handles the accessing (posting) of messages, the referencing of data, the scheduling of events (in response to subsystem or special events) and has access to the simulation clock time. SIMGEN merges the existing ASSM and the SDF into an executable simulator of either variety, depending on the option chosen when the function is executed.

For this evaluation, BETA simulations were performed on the C^3I and NELS subsystems of the ASE. A GAMMA simulation was performed on the NELS subsystem. These simulations were performed to evaluate the scope of the simulators and the utility of the results to an analyst.

For the C^3I subsystem of the ASE, 4600 lines of RSL were used for the ASSM of which approximately 3100 lines were actually

used by SIMGEN, (the others being originating requirements, descriptions, etc.). Included in the 3100 lines were 186 lines of BETA definitions. The SDF required 286 lines of Pascal. SIMGEN itself generated an additional 5784 lines of Pascal.

4.2.4 Design of a System

The specifications ([ASE80] and [CSID80]), and the ASSMs produced using SREM, were both used as design statements (Volume II, Appendix B of this document contains an example). This was done to further identify the utility of SREM within the Air Force life cycle. The quality of the designs was assessed using several design evaluation techniques including design walkthroughs, Myers' measures (Volume II, Appendix G), McCabe's measure (Volume II, Appendix G) and the design characteristics of hierarchical structure and levels of abstraction. The results of using these techniques produced a comprehensive view of the merits of each design.

All designs, those produced directly from a specification document and those produced from an RSL specification, were represented by the Martin Marietta Visual Control Logic Representation (VCLR) software design method [MMSE79]. For all practical purposes, a VCLR is equivalent to a Nassi-Schneiderman diagram.

VCLRs allow neat and concise representations of structured logic paths such as IF-THEN-ELSE, DO WHILE, DO UNTIL and DO CASE constructs (Fig. 4-1). The goal of a VCLR is to present the logical organization of modules, along with control, and data flow rather than reflect an implementation (code level) description. For example, when a lower level module is referenced in a higher level VCLR, an explicit "call" of that module is not necessarily implied. The implication is that the

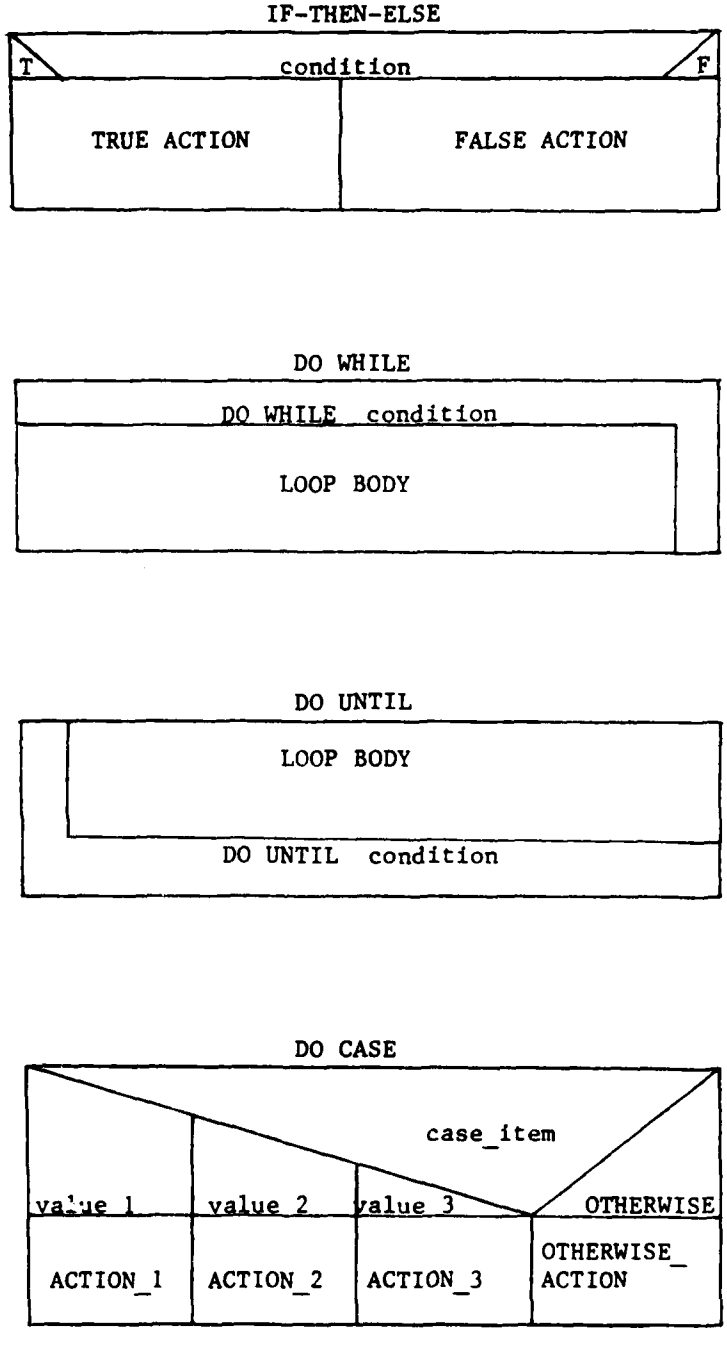


Figure 4-1
Sample VCLR Constructs

lower level module is to be executed at the point in processing where it is referenced in the higher level VCLR.

To make possible direct comparisons between designs produced from different requirements sources, some guidelines were set for the design efforts. First, the specification was to be followed precisely in that all requirements were to be fulfilled. Second, each design was to be accompanied by a list of assumptions made by the designer, i.e., decisions made to clarify ambiguities. Finally, the Martin Marietta standards for VCLRs were to be followed as closely as possible without making the design logic too complex or obscure. For example, the maximum number of branches allowed in a CASE construct was simplified by Boolean analysis because one CASE construct in the CSID design required 14 branches. These guidelines helped to ensure consistency in the presentations and reviews for each design.

4.2.4.1 Comparison Techniques

To provide objective comparisons between the designs derived from the two different specification techniques, it was essential to develop a set of desired characteristics and a series of measures to determine the degree of existence of those characteristics. The design characteristics used for the comparison (based on the principles described in [ROSS75]), were chosen because they are characteristics whose presence or absence can be readily determined in an objective manner.

4.2.4.1.1 Design Walkthroughs

A walkthrough of the design logic is the first

method employed to evaluate designs for adherence to specifications. During the walkthrough, any exceptions are duly noted and the designer is given an opportunity to answer any questions. The designer is then responsible for correcting the design immediately following the walkthrough. This process is repeated, if necessary, until the design is accepted.

The analysts who participated in the walkthroughs were all familiar with the VCLR standards, the requirements for the particular system being designed and the walkthrough techniques. The final versions of the designs for the NELS subsystem of the ASE are shown in Appendix B of Volume II of this report.

4.2.4.1.2 Hierarchical Organization

A hierarchical structure allows a system design to be divided into different levels of understanding [MYER76], postponing unnecessary levels of detail. Each level represents relationships among the parts of the lower levels (Fig. 4-2). This method reduces the degree of complexity that the designer must handle at any one time.

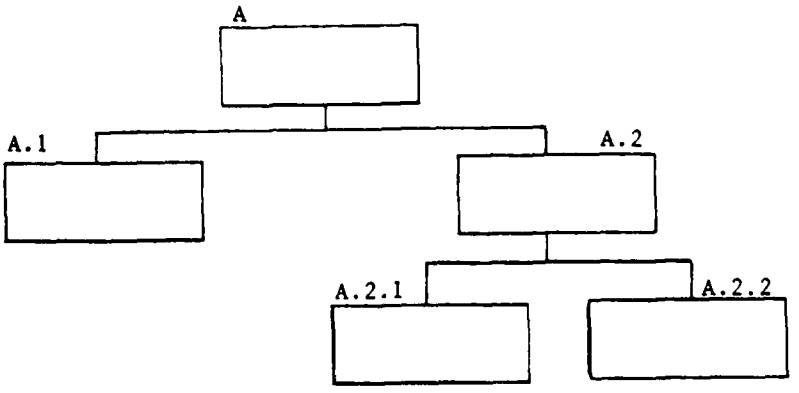


Figure 4-2 Hierarchical Structures

The designs produced directly from the specifications were done in a hierarchical manner, following the Martin Marietta software design standards [MMSE79]. In a loose sense the RSL designs can be viewed as supporting a hierarchical structure, (Fig. 4-3), if one considers that the R_NET's and SUBNET's represent processing levels, and there is no restriction on the detail of processing that any ALPHA may describe. However, SREM is a data flow methodology, as is evident in the restriction of single entrance, single exit placed upon the processing constructs (SUBNETs and ALPHAs). Note in Figure 4-3 the ease with which a "data element" (GRASS_CLIPPINGS) could be associated with the input and output interfaces. The importance placed upon data necessarily increases the level of detail required during the definition of the processing steps (ALPHAs). SREM-described systems should be regarded as having minimal hierarchical organization.

There is a stronger hierarchy concept associated with SREM-described information structures. This is enforced in two areas, data structuring and RADX processing of user defined sets. In the latter case, a structure can be established for specifying the order to extract and display information. In this case a hierarchy is explicitly defined as a connected graph and establishes a trace path through the structure. It is unfortunate that this vocabulary is part of the REVS terminology because a hierarchical transaction path of RADX processing or hierarchical data definitions may impute hierarchical organization to the target system.

4.2.4.1.3 Levels of Abstraction

```
R_NET:  YARD_WORK.
    INPUT_INTERFACE:  HOME_OWNER
    STRUCTURE
         DO
              ALPHA TRIM_THE_HEDGE
              TERMINATE
         AND
              SUBNET  MOW_THE_LAWN
              TERMINATE
         AND
              ALPHA  FIX_THE_HOSE
         END
          .
          .
SUBNET  MOW_THE_LAWN
    STRUCTURE
         DO
         ALPHA  SHARPEN_THE_BLADE
         AND    CHECK_SPARK_PLUG
         TERMINATE
         DO
              ALPHA CUT_GRASS
         END
          .
OUTPUT_INTERFACE:  GRASS_GLIPPINGS_BAG
          .
          .
          .
```

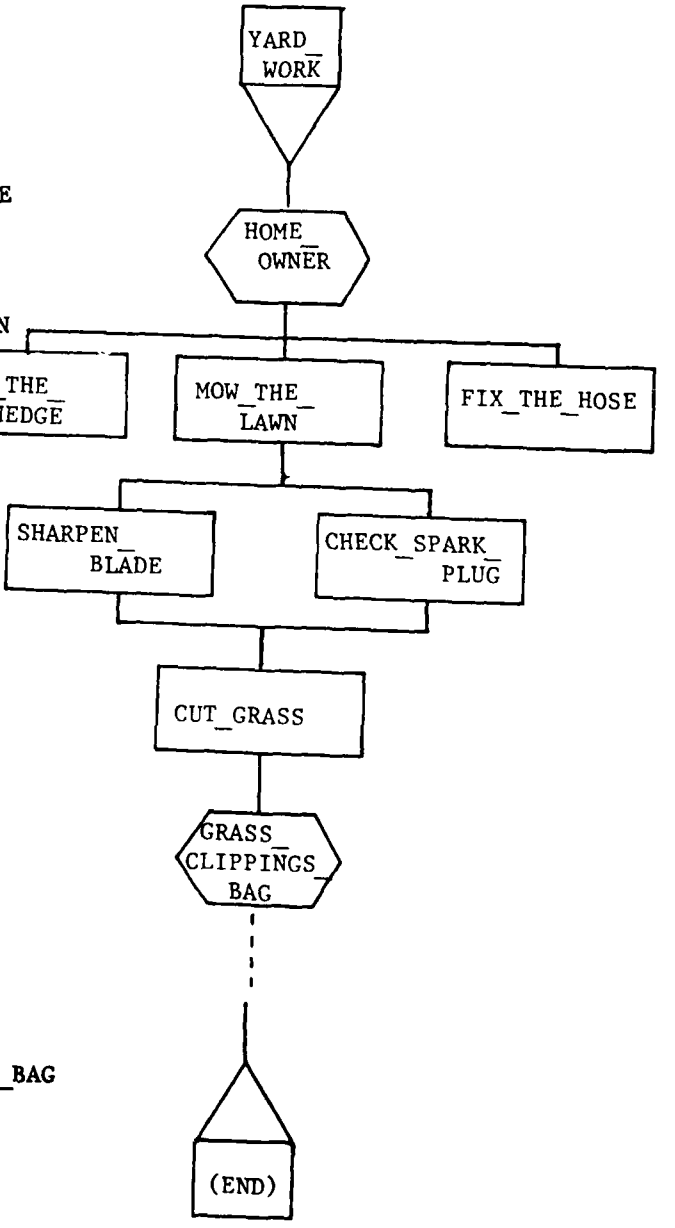


Figure 4-3 RSL Hierarchical Structure

Abstraction within a software design can be viewed from two distinct perspectives -- procedural abstraction and data abstraction. The idea of procedural abstraction is closely related to that of functional decomposition. Each level of decomposition represents a corresponding degree of detail, or abstraction, in the description of the processing the system must perform. The design is formed in a series of stepwise refinements as each procedure level becomes more specific and detailed.

Data abstraction can be viewed as the complement of procedural abstraction as an approach to software design. The idea of data abstraction is to identify all the properties of data objects, thus identifying the characteristics of things before describing the procedures that manipulate them. This technique allows for a stricter definition of how data will be used and results in a set of procedures and descriptions that together provide less ambiguous and more controllable data designs.

With the focus of SREM on data flow modelling, it supports data abstraction very well. Because ASSM analyses concentrate on the completeness and consistency of data description and data flow, data abstraction as a design technique is more readily accomodated by the SREM approach than procedural abstraction.

4.2.4.2 Derivation of Design from Specification Document

The generation of designs using the [ASE80] and [CSID80] specification documents as the source of

information was pursued in a top-down manner using VCLR techniques. A walkthrough of the system was performed after each level of the system (functional decomposition) was described .

As each level of description in the specification was broken down into its constituent parts, each part, or group of parts that formed a logical segment became a function. The information contained in both the CSID and ASE specifications greatly aided the functional decomposition process. The CSID specification included a Program Design Language (PDL) description complete with detailed functional blocks. Lower level blocks were constructed by grouping processing statements under a logical name describing the function being performed. Similarly, the ASE specification facilitated the breakdown of processes with its sequence of DeMarco diagrams. These began by showing the overall system (at a high level) and were followed by a more detailed diagram of each subsystem. The textual descriptions followed a similar pattern.

Other than the modules, the primary software items present in the CSID and ASE specifications were data items and files. In most cases, the high-level descriptions of these items were sufficiently complete and consistent that they could be easily decomposed or even directly included in the designs.

4.2.4.3 Derivation of Design from RSL

A computer listing of all RSL elements contained in an ASSM was used as the specification document in the generation of a design from RSL. The original specification document was used only for clarification during a design

walkthrough and even then by someone other than the designer. On occasion, the reason for an RSL structure, or a particular portion of a structure, was not clear to the walkthrough participants. This lack of clarity was usually caused by insufficient textual description within the RSL listing; the specification document was used as the authority in such situations.

The designers were instructed to begin with the R_NET and SUBNET STRUCTUREs to get a feeling for the overall logic, processing and initial decomposition of the system. For more detail (and other information, like the purpose or function of an ALPHA), the designer used the alphabetical computer listings of all RSL elements with their associated attributes and relationships.

Because the R_NET STRUCTURE provides a logical data flow path description of a system, much of the logic in the RSL was directly and simply translatable to a VCLR representation. There was a conscious effort to use descriptive element names and provide meaningful descriptions of each element to keep the time spent searching the RSL listings for meanings to a minimum.

The more complex RSL nodes (Fig. 4-4) presented little difficulty in being translated from RSL to a VCLR construct. The OR and CONSIDER_OR nodes became IF-THEN-ELSE or CASE constructs, depending on the number of branch points. The FOR EACH node was mapped into a DO WHILE construct.

An AND node became a sequential process. Since the order of processing is not specified by an AND node, an assumption was made as to which branch was to be processed

first. If an AND node required parallel processing (documented in RSL text fields), the processes were still sequenced, because there is no practical way to represent parallel processing with VCLR constructs. This generally resulted in some logic changes in other parts of the system, in addition to forcing some assumptions about queueing up messages for input into a process.

4.3 Results

4.3.1 The Life Cycle

In section 2.3.1, the definition of the software life cycle used in this document was given. According to the evaluation project's experience, it appears that SREM and its accompanying tools (REVS) best support the activities that take place part way through the requirements and into the system design phase. This must be further qualified by stating that SREM is specifically targeted for use after requirements have been allocated to functional system software components but before detailed design and processing algorithms have been defined.

4.3.2 The Methodology

As stated in the SREM reference materials, the methodology intends to be a means of defining software requirements to produce a functional specification for the software components of a system. The evaluation project's experience, particularly with the ASE system, has shown that ambiguous and conflicting requirements must be resolved before RSL can be properly used to describe a system. Use of the SREM methodology does in fact greatly aid in identifying ambiguous requirements. The following tabulation shows the classification of specification errors, the

```
CONSIDER DATA ASET_MSG_NAME_DATA
IF (C3I_TEXT_MESSAGE)
   SUBNET C3I_SUB_1
OR (UNIT_GROUP_ID_REPORTS)
   SUBNET C3I_SUB_2
OR (CANDIDATE_TASK_RESPONSES)
   SUBNET C3I_SUB_3
END
```

ASET_MSG_NAME_DATA			
C3I_TEXT_ MESSAGE	UNIT_GROUP_ ID_REPORTS	CANDIDATE_ TASK_RESPONSES	OTHERWISE
DO C3I_SUB_1	DO C3I_SUB_2	DO C3I_SUB_3	ERROR HANDLER

```
FOR EACH ENTITY_CLASS DYNAMIC_DISPLAY SUCH THAT
         (NODE_NUMBER = FRAME_NODE_NUMBER) DO
         ALPHA COLLECT_TEXT
END
```

ACCESS FIRST DYNAMIC_DISPLAY INSTANCE	
DO WHILE DYNAMIC_DISPLAY INSTANCES LEFT	
T NODE_NUMBER = FRAME_NODE_NUMBER	F
COLLECT TEXT	NULL
ACCESS NEXT DYNAMIC_DISPLAY INSTANCE	

```
DO
  SUBNET C3I_SUB_1
AND
  SUBNET C3I_SUB_2
END
```

SUBNET C3I_SUB_1
SUBNET C3I_SUB_2

Figure 4-4

Sample RSL-to-VCLR Mappings

Error Class	Percent of Total	Method Found
Inconsistencies	36	80% analyst 20% REVS
Ambiguities	21	100% analyst
Incompleteness	43	100% analyst
Algorithm flaws	N/A	N/A
Total	100	

percentage of the specification errors in each classification and the percentage of each error that each method (found by REVS or found by an analyst using the SREM methodology) uncovered.

It is the evaluation project's contention that requirements errors uncovered by SREM occurred more as a result of the application of its strongly disciplined approach and not because of the analytic power of its tools.

4.3.3 Information Content

The discussion of the information content within SREM and its related software tools can be divided into three topics that are aligned with the three major sources of information, namely; the SREM discipline, REVS and RSL. Each of SREM, REVS and RSL attempts to capture information on the different aspects of the software requirements development process they support. While it is an overall SREM goal to capture information on all aspects of the software requirements development process, sufficient limitations exist to restrict the application of SREM to specific function and data definition activities.

SREM can be used to provide management information on the software requirements development process. To accompany the distinct phases of the methodology, the tools and procedures can be used to provide derived management control and schedule

information. The information captured relates to where in the process of requirements analysis a project finds itself. This is expressed in terms of the current status of the requirements analysis and the criteria that determine when one can progress to the next phase.

REVS provides information about the contents of an ASSM. This information depends on the phase of SREM application currently in process and the contents of the ASSM used by REVS in producing its information. The information in a report has a broad range; the RSL report contains syntactic and semantic information on the ASSM elements, RADX has attribute, relationship, completeness and consistency information, and the SIMGEN contains information on the structure and information flow of the system. The report contents are produced by performing set analysis on the collection of objects and relations between objects in the ASSM.

In RSL, one describes both the stimuli that put a software system in motion and the reaction of the system to those stimuli. The stimuli are described in terms of RSL messages that describe some data object in the software system. When one of these messages is created, the RSL requirements network (R_NET) is invoked to respond to the message. The R_NET responds to the message, or class of messages, by creating or deleting data in the system, creating or deleting messages in the system or by invoking other R_NETs. The information content of RSL is more easily visualized by treating an RSL description as a model of a system in terms of a finite state machine. Using this representation, messages become the stimuli that provoke state transitions and R_NETs become the processing that occurs during a particular state change.

4.3.4 Maturity of SREM

Maturity refers to the level of development of the concepts supported by a system. That is, have the concepts arrived at some semblance of a "standard", or are they still in the process of being refined?

SREM is a data flow methodology rather than a hierarchical decomposition methodology. Instead of organizing requirements into a hierarchy of functions, SREM uses the data input, process, output approach. This approach to describing software systems is well defined and has the attraction of being specific enough in its descriptions to be understood, yet general enough to handle many processing situations. This is an appealing method to use in high-level system specification descriptions.

A SREM-described system can be considered as a finite state machine description of the software being developed. Finite state theory is widely taught and accepted as a means of describing software processes in a manner that allows mathematical and logical analysis [HOPCR69]. Finite state automata theory states that all software can be described in a data flow manner, but in practice it is not always the preferred method of designing an application system because it does not provide an overall view of the system structure.

The concepts used to formulate the SREM methodology are mature and respected. However they do require that the target system be modeled, and a model is at best an approximation of what is desired. The strength of this approach is the ability of humans to cope with and communicate definitive system information on a more abstract level than is needed for machine implementation. Consequently one should not expect SREM, or any other high level specification scheme to precisely represent a system, but should expect that its expressive power is somehow proportional to its representation accuracy and remaining

ambiguity. While such an "information content" index cannot be rigorously derived, we believe that SREM would rate very highly against comparable systems today.

4.3.5 Usability of SREM

Usability refers to the practical mechanics of dealing with the SREM methods and tools from an human-factors standpoint. In particular, the evaluation team addressed the utility of SREM to a design group as an expression medium for intercommunication.

The management of a project using SREM for requirements definition and analysis needs some means of monitoring progress as the requirements definitions become more established. The RSL elements SUBNET and R_NET provide natural boundaries from which to gauge work loads, time spent and progress made, but there are no specific elements or attributes in the basic RSL requirements definition set that qualify the progress. The language extension function of REVS provided the evaluation team with new attributes that aided in progress qualification. For example, the new attribute DATE-ENTERED matched well with the attribute ENTERED_BY to increase information on time spent and progress made. The new attribute JUSTIFICATION allowed textual information on evolving or changed requirements to reside within the ASSM data base for future reference or evaluation. It should be noted that the project manager must understand and be aware of all RSL terms, constructs and extensions.

A requirements tool used on a large project should aid the organization of requirements and the communication of requirements between analysts. The ASSM provides a centralized location in which analysts may find information concerning previously defined requirements. Caution should be exercised when merging analysts' work with an existing ASSM to ensure that

single elements do not have multiple names and hence add confusion. For example, an analyst whose responsibility is to define subsystem B, creates the following message to be output from one subsystem (B) to another (A):

```
MESSAGE  data_to_sub_a
     MADE BY
          DATA a
          DATA b
          DATA c
```

The creation of this message implies that the analyst has determined the need for three items (a, b, c) to be sent to subsystem A. Another analyst, who is responsible for defining subsystem A, creates the same transaction as a message from subsystem (B) to subsystem (A):

```
MESSAGE  data_from_sub_b
     MADE BY
          DATA A_123
          DATA XYZ
```

This message implies that the analyst for "A" interpreted the need for two data items (XYZ, A_123) to be received from subsystem B. Both analysts are conceptually creating the same message but have identified the contents much differently. If these MESSAGEs are not screened prior to their insertion in an ASSM, there will be information in the ASSM that is inconsistsent.

Naming conventions for RSL elements have a favorable impact on maintaining control as the project size increases. The following naming conventions were used by the evaluation team during the specification translations of the ASE subsystems. The consensus of the team was that the conventions greatly aided maintaining control over the data requirements that were input. Determining whether to use a prefix or suffix depended on the uniqueness requirements of the names as well as what made sense when being read.

ELEMENT TYPE	SUFFIX	PREFIX
ALPHA	_ALPHA	--
DATA	_DATA	--
DECISION	--	DECISION_
ENTITY_CLASS	_EC	--
ENTITY_TYPE	_ET	--
EVENT	_EVENT	--
FILE	_FILE	--
INPUT_INTERFACE	--	INTO_
MESSAGE		
INPUT	_MSG_IN	--
OUTPUT	_MSG_OUT	--
ORIGINATING_		
REQUIREMENT	--	ORIG_REQ
OUTPUT_INTERFACE	--	TO_
PERFORMANCE_		
REQUIREMENT	--	PERF_REQ
R_NET	_R_NET	--
SOURCE	--	SOURCE_
SUBNET	_SUB	--
SUBSYSTEM	--	SS_
SYNONYM	--	--
UNSTRUCTURED_		
REQUIREMENT	--	UNSTRUC_REQ_
VALIDATION_PATH	_VAL_PATH	--
VALIDATION_POINT	_VAL_POINT	--

4.3.6 ASSM Analysis Results

Analysis of the ASSM data bases proceeded in standard RADX fashion by using the set/subset partitioning concept. Conceptually, a RADX set is a named collection of elements in the ASSM. Predefined sets are directly referenced when RADX is activated and consist of the universal set (ALL or ANY of the elements in the ASSM), element type sets (all elements of any named type), and basic element sets (enumerated elements). Additional subsetting can be achieved by user definition of each set and the conditions for membership in it.

User-defined sets were created by the evaluation project to analyze the ASSM by eliminating all the redundancies of the predefined RADX sets. They were necessary because often the analyst was not able to choose predefined sets which provided the

information desired for an ASSM analysis and because of the inefficiency of the predefined sets. In comparing the results of the user-defined sets and the results of the predefined sets, the consensus among the analysts was that the user defined sets were as complete and as usable as the predefined sets. Because the user-defined sets required substantially less computation time (in terms of both computer connect time and computer processing time) and revealed all of the requested information on ASSM element completeness and consistency, they were considered to be more flexible and applicable to an evolving ASSM.

RADX can easily check for the existence or nonexistence of attributes or relationships belonging to an ASSM element and also distinguish between single and multiple occurrences of relationships. Hence, the completeness and consistency of an ASSM is relatively easy to check. Elements contained in an ASSM but not associated with any RSL structure (considered inconsistent) can be called to the attention of an analyst, as can instances of elements that have one or more of their necessary attributes or relationships missing (considered incomplete). An example of an incomplete element would be a MESSAGE that PASSes through an interface but is never FORMed by an ALPHA.

The interpretation of RADX output is not always straightforward. Any incompleteness or inconsistencies within the ASSM can be a result of one or more of three causes:

1) An inherent limitation or inability of RSL to adequately describe a requirement, for example, parallel processing;
2) An incomplete or inconsistent ASSM. The analyst has not finished creating the ASSM or has made an error;
3) An incomplete or inconsistent software requirements specification.

It should also be realized that an erroneous ASSM caused by an erroneous software requirements specification will, in most cases, pass all of the RADX-phase analyses. For example, a MESSAGE may be MADE, FORMed and PASSed through an OUTPUT_INTERFACE, but a corresponding input message may never be received by another subsystem in the same ASSM. Neither this, nor its converse (MESSAGE received, but never sent) can be detected by RADX. Likewise, RADX was not designed with the intelligence to detect nonsense processes such as the requirement to explore the planet of Pluto.

The most disconcerting problem found during the ASSM analysis was within the VAX implementation of the RADX data flow analysis. The ASE specification required the C^3I subsystem to handle many messages from various input sources, filter these messages for errors and route the messages to the appropriate process. The data flow analysis would not accept the number of messages (and hence the volume of data elements) passing through ALPHAs that were created for the filtering of erroneous messages despite the fact that the requirements stated that the system deal with the messages in this manner. The resulting RSL can be seen in Figure 4-5. The data flow analysis indicated that too many messages were passing through one ALPHA (the ALPHA CHECK_THE MESSAGE filter in Fig. 4-5). This error indication is completely unacceptable when dealing with the ASE system, or with systems with like requirements, because the requirement is both reasonable to do in terms of requirements, design and coding, and can be described in RSL structure.

4.3.7 Simulation Results

The building of the Simulation Driver Definition file (SDF) for the C^3I subsystem was a relatively simple operation. The SDF is merged, by the SIMGEN function, with predefined variables

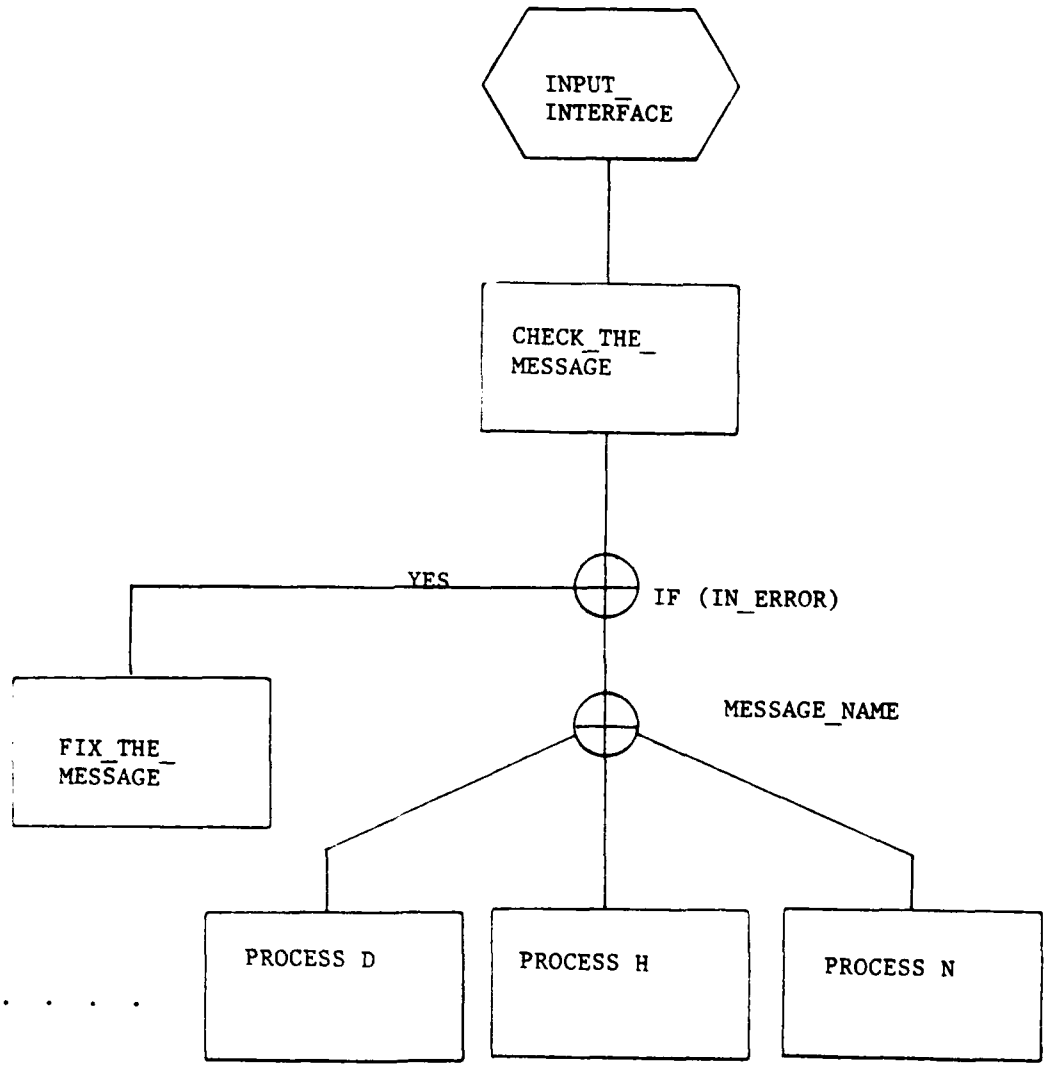


Figure 4-5

RSL Translation of a Requirement

and types from the ASSM to create a simulator. The building of the simulator became difficult when it was discovered that the rules on name uniqueness for RSL and the simulator are quite different. Names used in RSL must be unique within the first 10 characters, but the simulator demands that name uniqueness occur within the first 6 characters. Because these restrictions resulted in the need to rename the majority of the ASSM element names to be used in the simulation to make them unique to six characters, the evaluation team decided to rename all of the elements to keep a logical and consistent naming convention throughout. The simplest solution, and the one employed, was to add a prefix to the names of the elements. The prefix consisted of a letter followed by three numbers. For example, ALPHAs were prefixed A 001 through A_999 and DATA was prefixed D_001 through D_999, so the ALPHA FILTER_THE_MESSAGE became A_001_FILTER_THE_MESSAGE.

The results of the simulator executions (SIMXQT) consisted of a computer listing indicating the path traveled through an R_NET by a MESSAGE. The simulator clock time was indicated first, the message name and the network name appeared next, and then a listing of all the nodes within the network activated by the message appeared. Any analyses of these timing and path trace results must be performed manually. The practicality of generating a "simulation" only to ultimately perform a manual evaluation of its results is a questionable capability.

4.3.8 B-Level Specification

A MIL-STD 490 Type B-5 requirements specification document contains four sections that are directly applicable: Scope, Applicable Documents, Requirements and Quality Assurance. The Requirements section has four subsections: General Requirements, Program Definitions, Functional Requirements and Data Adaptation. The SREM components appropriate to the Requirements section are as follows:

Section 3 Requirements

3.0 General Requirements

ORIGINATING_REQUIREMENTs

DECISIONs

3.1 Program Definitions

INPUT_INTERFACEs

OUTPUT_INTERFACEs

MESSAGEs

R_NETs

3.2 Functional Requirements

SUBNETs

ALPHAs

3.3 Data Adaptation

ENTITY_CLASSes

ENTITY_TYPEs

DATA

FILEs

Section 4 Quality Assurance Provision

VALIDATION_PATHs

The computer generated output of SREM (Volume II, Appendix D) is fairly readable except for the inescapable clutter caused by RADX echoing every command on the output document. Properly formatted these could provide a sort of header for the information that follows.

A problem arises when one considers the amount of information that must be manually added to the information output by RADX to complete an entire B5 specification. The additions to the RADX information include the section headers, the introduction paragraphs for each section and subsection, and the textual information between the subsections that is necessary to make them cohesive.

The question of understandability of RSL by the people who would be involved with utilizing a B-5 specification based on the RSL immediately arises. Because there are some potentially ambiguous constructs in RSL (e.g., the FILE, ENTITY_CLASS AND ENTITY_TYPE element types), it is quite possible that anyone without significant RSL experience would interpret them incorrectly.

The basic information content of SREM covers only two sections of the B-5 document (Section 3 Requirements and Section 4 Quality Assurance Provision). The evaluation team considers the amount of information which must be manually added to the ASSM to generate the entire B-5 document too great to practically utilize RADX for fully automating B-5 specification generation. Also, the information which must be added to the ASSM is sufficiently unique to make it impractical to include as a permanent extension of RSL.

4.4 Recommendations

4.4.1 The Life Cycle

It is recommended that SREM be used in the life-cycle phases of Software Requirements Definition and Design. SREM has proved to be of great utility throughout the entire Requirements Definition phase by making available element types, attributes and relationships which can describe software requirements to varying degrees of detail. In the design phase, SREM was most useful during the initial steps of identifying proper processing sequences and conditions. Therefore, the evaluation team recommends that SREM be used throughout the Requirements Definition phase and during the preliminary stages of the Design phase.

The usability of SREM in relation to managerial concerns of project control was not a detailed level objective in its development. Instead, SREM stresses the higher level, recognizable completion criteria of its steps and phases to establish control, and incorporates methods to record requirements traceability and open issues. The centralized ASSM data base provides a strong usability focus for requirements analysis, which can be greatly enhanced by using language extensions for the description, analysis and clarification of requirements information, and by establishing naming conventions. Specifically, the "DATE_ENTERED" and "JUSTIFICATION" constructs add control and tracking data. As in any design activity, the early establishment of naming conventions, which should be formalized beginning in SREM Phase 1, contribute to interface consistency.

4.4.2 User Interface (REVS)

The recommendations concerning the REVS user interface are divided into segments according to the SREM functions of RSL, RADX and SIMGEN.

4.4.2.1 Language Translation (RSL)

The recommendations concerning RSL include improvements to the translator and the interface between the user and REVS. It is recommended that RSL be extended to enhance the AND construct to emphasize parallel processing capabilities. By including a keyword IN_PARALLEL, the processing requirements would become more specific and the changes required to the language would be small although there would be some impact in the simulation software. It is also recommended that the language be permanently extended to allow for the inclusion of such managerial

concerns as metric measure values, statistical weightings, completeness values, etc.

It is recommended that the RSL translator be modified to include limitations on the numbers of translation errors based on a maximum upper error limit and on the severity of errors detected. Each RSL element attribute, relationship and structure should be given an error severity number that correlates to the degree to which an error of that kind would impact the translation process. For example, an error of the element attribute DESCRIPTION would have less effect on the translation, and hence a lower severity number, then an error occurring in the STRUCTURE of an R_NET. The RSL translation process would stop when (1) the total number of errors exceeds the error limit number, or (2) the number of severe errors exceed the severe error limit number. Both the error limit and the severe error limit numbers should be RSL parameters that can be changed by the user.

A complete rewrite is needed of the error detection and posting scheme to include informing the user of errors by specific textual messages in place of the current error codes. The ability of the translator to recover from errors that occur while processing an element requires major improvement. Further, it is recommended that a 'syntax check only' option be added to the translator. This function would take the input to the RSL translator and perform a normal translation, but would not create or modify an ASSM.

At the user interface level, an addition to the REVS executive language to include a HELP command is recommended. This command should be available from anywhere in REVS during an interactive session and contain specific

information about each REVS function. For example, a primer describing each REVS function (RSLXTND, RSL, RADX, SIMGEN, SIMXQT, SIMDA) in the form of function constructs and executable keywords with descriptive text, would be extremely helpful, especially just prior to the execution of the function.

A text editor capable of altering ASSM information would also decrease the time spent in modifying an ASSM. Currently, textual and structural ASSM information can be changed only by a complete removal of the element and then a reentry of the corrected element. For large descriptions or descriptions that contain small errors or for large network structures with only a few errors, the text editor capability would expedite the corrections.

4.4.2.2 Data Extraction (RADX)

The predefined sets that comprise the data analysis function should be increased to include sets that reflect managerial concerns. These sets should deal with such information as metric measures, completeness values and statistical weightings. It is recommended that the set construction function be modified to include combining multiple selection conditions. For example, a legal construction should look like this:

```
SET OK = ALL WITH TRACED OR (ALL WITH DESCRIPTION AND
SET                          GOOD_ONES).
```

This construction is more readable than the two separate sets that presently must be created:

```
SET B = ALL WITH DESCRIPTION AND SET GOOD_ONES.
SET OK = ALL WITH TRACED OR SET B.
```

4.4.2.3 Simulation Generation (SIMGEN)

For a smooth transition from RSL to SIMGEN we recommend that the ASSM element name uniqueness be equivalent to the identifier uniqueness rules imposed by the Pascal compiler used.

Because of the computer resources required to run a simulation, and the personnel resources required to evaluate the results obtained from the SIMXQT function output, it is recommended that the BETA simulation function be used only to demonstrate the feasibility of the defined interfaces.

GAMMA simulations, to be informative, require complete implementation level elaboration of function and data. This level of detail is inconsistent with the requirements analysis activity appropriate for SREM. The degree of detail necessary to construct a useful GAMMA simulator excludes it from being recommended for use.

4.4.3 System Development

4.4.3.1 Personnel

Considering the experience of this project in using SREM on two USAF software requirements specifications, and having available a diverse set of project personnel, there are two recommendations for staffing a project that will employ SREM. First, we recommend that all persons assigned to the project have, as a minimum, skill levels that include a substantial previous computer knowledge. They should be familiar with hierarchical structures, modularity concepts, design concepts and such relationships between assigned

variables as data, files, records and variable typing. These skills translate to approximately the level of a requirements analyst. The starting point of the SREM training session is at such a level that a lack of these skills would significantly lengthen the learning curve.

SREM, as a development tool, has proved to be an adequate cataloger of requirements information and relationships. This feature leads to the recommendation for limiting the number of people involved in using SREM for a project. While any specific number or ratio depends on the size or scope of the project, the centralization and accessibility of information that SREM provides results in the need for fewer analysts because most of the project administration, bookkeeping and version control is done by the SREM tools.

4.4.3.2 Computer Resources

The evaluation project staff believes REVS execution should be done in a batch environment, because the majority of the processing on the information residing in the ASSM does not require any interactive user actions. The batch environment recommendation will also preclude capturing the terminal as a resource for long periods of time.

5.0 QUALITY OF AVAILABLE SREM TRAINING

The SREM training course was evaluated from two different frames of reference. First, the course was examined with respect to its own stated objectives in terms of the knowledge it was intended to impart. The course was also evaluated in terms of audience benefits, i.e., did the course impart knowledge in a way the students found useful and comfortable?

5.1 Goals

To determine if the intended knowledge was imparted in an effective manner, the SREM course structure, presentation and materials were evaluated. The logical structure of the course was analyzed in terms of:

1) Audience considerations - What was the target audience and did the course address its needs? Was the amount of material covered reasonable in the time allowed?
2) Course overview - Were the objectives of both SREM and the training made clear?
3) Planned activities - Did the class exercises support the material covered in lecture? Was sufficient hands-on experience included?
4) Progress evaluation - Were criteria given so students could evaluate whether they had learned the material? Were the criteria related to observable student behavior?
5) Course summary - Was provision made for summarizing or reviewing difficult concepts?

Training presentation was examined with respect to:

1) Teaching methods - Was the quality and amount of

instructor-student interaction adequate? Were discussions encouraged? Were examples used to clarify difficult concepts?

2) Environment - Were the conditions under which the course was taught comfortable and conducive to learning?

Materials were evaluated in terms of how well they followed the lesson seq· ·ce, the degree of cross-referencing, their use of examples and their ease of use. Every attempt was made to evaluate the course itself rather than the instructor-dependent qualities of the course.

5.2 Methods

Before detailing how the SREM training course was evaluated, a description of the actual course is in order. The training session attended for this assessment was slightly altered to accommodate the large percentage of attendees who did not intend to be SREM users but were interested in the methodology. The actual course was presented as follows:

Week 1 - 40 hours of lecture with viewgraphs, class work on examples and some group discussion;

Week 2 - 40 hours of applications exercise, including informal discussion, instructor assistance in analysis and preparation of materials for exercise and experience in submitting, correcting and analyzing the RSL description of the example;

Week 3 - 40 hours of lecture, viewgraphs, informal discussion and applications exercises in preparing simulation models.

The total 120-hour course was divided into an initial two week course on SREM, RSL and RADX. The third week, presented at a later time, pertained to the SREM simulation models.

To achieve the stated objective of transferring software requirements engineering technology, the SREM training course proposes to impart the following information [PASI80]:

1) An overview of what SREM will do;
2) The facets of SREM applicable to project management;
3) The steps involved in the Software Requirements Engineering Methodology (SREM);
4) Features of REVS and RSL that are applicable to steps in the methodology, where they are applicable and what they do for the requirements engineer;
5) Hands-on experience through examples and homework.

After completion of the first two weeks of the course, students were polled for their reactions. Figure 5-1 is a sample of the survey. In addition to the survey, several attendees, including professional educators, submitted their individual evaluations of the course. The final assessment is a synthesis of these results.

5.3 Results

On completion of the SREM training course, review of the student survey results (Table 5-1) and review of the individual evaluations, the training effectiveness was assessed. Observations pertaining to the course structure, presentation techniques and materials are covered in this section.

5.3.1 Course Planning and Structure

NAME LAST FI

DEPT. - SECTION

LABOR GRADE

HOW LONG DID YOU ATTEND THE COURSE (WEEKS)? |1|2|

HOW COMPETENT DO YOU FEEL YOU ARE IN WRITING SYSTEM REQUIREMENTS? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... NOT COMPETENT AT ALL
5 ... COMPETENT
9 ... SUPER COMPETENT

HOW EXTENSIVE IS YOUR UNDERSTANDING OF REQUIREMENTS DEFINITION TOOLS? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... NO UNDERSTANDING
5 ... GOOD UNDERSTANDING
9 ... EXTENSIVE UNDERSTANDING

HOW DO YOU FEEL RIGHT NOW ABOUT USING SOFTWARE REQUIREMENTS DEFINITION TOOLS ON THE JOB? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... NEGATIVE FEELING
5 ... AMBIVALENT
9 ... VERY POSITIVE FEELING

HOW WOULD YOU RATE THE AMOUNT OF INSTRUCTOR INTERACTION WITH THE STUDENTS? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... LESS THAN ADEQUATE
5 ... ADEQUATE
9 ... VERY ADEQUATE

HOW WOULD YOU RATE THE QUALITY OF INSTRUCTOR INTERACTION WITH THE STUDENTS? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... POOR
5 ... GOOD
9 ... EXCELLENT

HOW WOULD YOUR RATE YOUR ABILITY TO APPLY WHAT YOU LEARNED IN THE SREM COURSE? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... NOT AT ALL
5 ... SOME
9 ... COMPLETELY

HOW WOULD YOU RATE THE OVERALL QUALITY OF THE COURSE? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... POOR
5 ... ADEQUATE
9 ... EXCELLENT

HOW WOULD YOU RATE THE QUALITY OF THE HANDOUTS AND VISUAL AIDS? |1|2|3|4|5|6|7|8|9|

SCALE: 1 ... POOR
5 ... GOOD
9 ... EXCELLENT

Figure 5-1
SREM Class Questionaire

Table 5-1. Training Survey Results

QUESTIONS	RESPONSES	MEAN
How competent do you feel you are in writing system system requirements	7*,7,1,6,5,6*,5,3,9,8,7,5*,5.25*	5.70 ± 2.09
How extensive is your understanding of requirements definition tools?	5*,7,2,9,5,7*,5,2,2,8,7,8*,6.25*	5.63 ± 2.40
How do you feel right now about using requirements definition tools on the job?	7*,8,5,9,6,7*,7,1,2,8,8,8*,7*	6.38 ± 2.40
How would you rate the amount of instructor interaction with students?	7*,6,5,4,7,6*,5,5,2,6,3,5*,5.75*	5.13 ± 1.45
How would you rate the quality of instructor interaction with students?	7*,7,4,3,6,6*,4,5,4,7,3,5*,5.25*	5.10 ± 1.44
How would you rate your ability to apply what you learned in the SREM course?	7*,2,1.5,6,6,7*,4,1,1,4,4,7*,6*	4.35 ± 2.34
How would you rate the overall quality of the course?	5*,6,5,4,6,5*,2,1,3,7,5,6*,4.75*	4.60 ± 1.71
How would you rate the quality of the handouts and visual aids?	4*,5,5,2,6,4*,4,1,4,8,3,5*,4.75*	4.29 ± 1.74

*Attended two weeks of the course

Because the course and lesson plans were not available, some of the following observations are to some degree conjectures based on observing the conduct of the course. Course planning and logical structure were analyzed in terms of the course overview, the pace of instruction, the intended audience, the division of the course into planned lessons with objectives and the techniques used in evaluating the students' progress.

While a brief overview of the subject of the course (SREM) was provided orally and is shown (in the SREM Management Overview [PASI80]) as being planned in the first lesson, a schedule at a very high level of detail was the only thing actually presented. A schedule appears on the first few pages of the SREM Management Overview, but was not used in this instance due to schedule changes and was not referenced to sections or page numbers within the course documents. Definitions of software life cycle and software requirements, the motivations behind SREM development, and the problems SREM addresses would have been motivational at this point, giving students a better picture of SREM's position in the development process. A brief description of SREM and its place in the software life cycle, plus a fairly detailed schedule, cross-referenced to a course guide, and containing lesson descriptions should have been provided at this time to give the students an initial overview of the course.

During the first week of instruction more than 700 viewgraphs (reproductions of text material) were presented, plus a number of additional slides. This covered more new information than one can expect students to learn in such a brief period. The excessive amount of material was partially caused by the change in the course schedule, but because the second week of

class was well filled with activities related to applying the information taught in the first week, it seems unlikely that a thorough treatment of SREM, including applications experience, can be accomplished in 10 days. The amount of information covered in the third week was more closely matched to the time available for instruction with sufficient schedule flexibility to allow for review and discussion.

The SREM course was intended to be taught to an audience of prospective users. In fact, the audience for the first two weeks of the course consisted of only a few people who intended to use SREM. The initial audience consisted of 18 people with software experience ranging from 0.5 to 24.0 years. Of these only three intended to be SREM users. The others attended because of a general interest in the methodology. To accommodate these two major student groups, the course format was changed from two weeks of alternating lectures and applications experience to one week of lecture, attended by all 18 participants, followed by a second week of informal discussions and laboratory work, attended by only four individuals. Despite this change in course schedule, the course objectives remained the same. For the third week of class, the original schedule and objectives were maintained but the audience changed slightly. Total attendance for the third week of class was six students, four of whom were directly involved in applying SREM to systems under development while the other two were attending to gain additional knowledge about SREM.

The attempt to accommodate both prospective users and those wishing a general knowledge of the methodology in the same course was only partially successful. The accelerated lecture segment, with no hands-on reinforcement of the things learned, resulted in

some frustration and confusion on the part of all of the students. The majority of the students really wanted a short course covering only the main points of the methodology. The details, presented over a week-long class, were not relevant to these people, but were part of the course material and were necessary for future users. Two separate courses would solve the problem. The audience for the third week closely matched the audience of SREM users for which the course was intended, and the problems associated with audience mismatch did not occur as they did in the first two weeks.

As far as the students could determine, the course was not explicitly divided into planned lessons with objectives, planned learning activities and achievement criteria for each lesson. The only apparent structure was based on the phases involved in applying SREM. The logical format of the course, while it mirrored the phases and steps of using SREM, was without clear transitions between logical units. This resulted in a somewhat confusing and haphazard sequence of presentation. The constructs being taught were unlikely to be strongly associated with the parts of SREM to which they belonged because there was no solid framework in which the student could place the new constructs such as would be provided by lesson overviews and summaries.

A minimal evaluation of learning was included in the course structure, and evaluation was not based on specified achievement criteria. During the first week there were occasional attempts at evaluation in the way of oral review of homework assignments and the asking (very occasionally) for information from the class (almost always without response). The second and third weeks contained no more in the way of criterion-referenced planned evaluation [MAGE62, MAGE67] than the first. However, the higher

level of instructor-student interaction and the feedback from the REVS software during the second and third weeks provided the students with a clearer understanding of their progress than in the first week.

5.3.2 Course Execution

While the consideration of course presentation may appear to be instructor dependent, many of these factors can be significantly altered by changes in course design. The factors of teaching methods used, student-instructor interaction and the environment in which the course was given are considered in this section.

One of the greatest problems with the teaching of this course was the choice of "viewgraphs in the darkness" as the main instructional technique. This method is known to have serious shortcomings, including: (1) a tendency to put students to sleep, (2) disruption and confusion resulting from the inevitable mishandling of the viewgraphs, particularly when attempting to retrieve a previously shown viewgraph from the stack for review purposes, (3) introduction of white noise from the projector fan, making comprehension of the lecture more difficult, and (4) reducing the level of instructor-student interaction. The viewgraphs used as lecture aids were mostly reproductions of pages in the SREM Management Overview. The total number of viewgraphs shown, and hence the total number of time spent in the dark, could have been reduced by referring the students to the appropriate pages of the SREM Management Overview. Too many viewgraphs were used, some shown for as little as 10 seconds. The viewgraphs were all black on white contributing to eyestrain and overall fatigue, were often too densely packed with

information, and sometimes contained sections that were washed out and difficult to read. The additional disruption caused by frequent interruption of the lecture to distribute handouts could have been avoided by giving an ordered set of handouts at the beginning of each lesson.

The teaching methods employed in the second and third weeks were much more effective than the pure lecture technique forced on the first week. They included short lectures followed by immediate application of the material just covered. Those students who attended the second week indicated on the survey a much greater confidence in their ability to apply SREM than those who attended only the first week even though all of the material had been covered in the first week. The levels of independent student activity and instructor-student interactivity was much higher than in the first week. More indepth review of difficult concepts such as ENTITY_CLASSes and ENTITY_TYPEs would have improved the second week, as would more information concerning the effective handling of the reports produced in doing a sample SREM application.

5.3.3 Course Materials

The course consisted of:

SREM Management Overview[PASI80] - Used as a textbook, reference and, to some degree, a course guide;

REVS Quick Reference User's Manual [LOSH79] - Used as a reference book;

REVS Users Manual [GUNT79] - Used as a reference book;

Requirements Development Using SREM Technology [ALF079] - An early version of [PASI80] but includes a section (Chapter 6) on Functional Simulation as a 1-week course

Viewgraphs - Used as visual lecture aids;

Handouts - Mostly consisting of forms to be used as worksheets in applying SREM.

The SREM Management Overview is not really a management overview, or even a text for a management overview. It is essentially a user's course textbook arranged in lesson order, interspersed with homework assignments. It was also used as the primary reference manual for SREM concepts and RSL constructs. The book contains a fairly complete description of SREM, with examples. However, it is too long for practical use as a text for a two-week course. It is too long and detailed for use as a course guide, and without a table of contents or index, insufficiently cross-referenced for use as a reference manual.

The degree of difficulty for examples presented in the SREM Management Overview was too simplistic in nature, e.g., they did not go beyond simple relations between processes. Only one example presented could show differing "angles of attack" that could be used in describing a problem in RSL. For instance, an example of a prompt-driven system would have been useful to show several effective solution approaches and may have promoted class discussion.

The text Requirements Development Using SREM Technology follows the same format as SREM Management Overview and has similar strengths and weaknesses. The primary use was in teaching simulation aspects which were not addressed in the latter document.

The REVS Quick Reference Manual provides a valuable, well-indexed reference to RSL syntax information and REVS constructs and operating procedures. Although it is a valuable tool when used in conjunction with the REVS User's Manual during SREM applications work, it requires too much prior familiarity with SREM to be an effective instruction tool.

The handouts were mostly worksheets for use in doing examples during the first week and in performing the application exercises during the second week. They were definitely of great utility during the second week, although some confusion resulted from duplication of function between some of the forms.

5.4 Recommendations

The following suggestions are descriptions of some important attributes of an improved SREM user's course. To be fully satisfactory, the course will have to be completely redesigned. This includes a thorough analysis of the tasks to be accomplished by the course and a target audience analysis, followed by the creation of criterion-referenced course objectives, learning activities to achieve those objectives, and evaluation activities to test achievement of course objectives. However, even without performing a complete re-design, specific recommendations can be made.

5.4.1 Improving Existing Course

1) When the students arrive at the first course session, they should be provided a written overview of the course. This overview should include a description of course objectives in terms of learned student behaviors, a brief description of the subject matter and scope of the course, and a course

schedule. The first session should include a period for discussion and questions concerning the points covered in the overview.

2) In addition to the course overview, the first day of the course should contain a subject matter overview. At least the following areas should be covered:

   a) The philosophies and types of problems addressed by software tools in general,
   b) The underlying philosophy of SREM in the context of software tools and their environments,
   c) A brief history of the development of SREM and its use on previous projects,
   d) Where SREM fits into the life cycle of a software system,
   e) The strengths and weaknesses of SREM in terms of such factors as project size and type of software being developed (i.e., real-time vs prompt-driven interactive systems),
   f) The overall structure of SREM, differentiating between RSL, REVS, RADX, ASSM and any other major structures and substructures, giving the main function of each.

   The material presented in the first day of the course could be expanded to become a short, high-level overview course for managers, customers and others who do not need to actually become skilled in using SREM. This first day of the course should also be used to promote instructor-student and student-student interaction. Questions concerning SREM's status as a requirements tool, the validity of the underlying philosophy of SREM, or any number of related

areas may arise, and free discussion should be encouraged. The atmosphere of the student involvement engendered by such discussions carries over into the rest of the course and is well worth the time taken from other activities;

3) The details of using SREM should be presented in a number of short lessons, each with its own overview, instructional objectives, learning activities and evaluation procedures based on criteria contained within the objectives. Because many of the problems associated with using SREM derive from SREM's unfamiliar terminology, each new term should be defined at the beginning of the lesson in which it is introduced. The material to be covered in the lessons should be chosen to emphasize the parts of SREM that a task analysis has indicated are especially important in using SREM. This might include requirements extraction, entity-class definition, message definition, net drawing, RSL translation and RADX interpretation. Additional emphasis should be given to concepts and skills that have been shown to be difficult for students to learn. The correct use of entity-classes and entity-types is a good example of a commonly misunderstood area in SREM that is of great importance in SREM applications;

4) The lessons should be sequenced and presented in such a way as to increase student participation in the learning process. This would be facilitated by sequencing the lessons in the same order as the various phases and steps of the SREM methodology that are applied in a project. To avoid introducing unnecessary complexity, the lectures or other presentations should be structured around the application of SREM to one sample system that, at completion

of the course, will have been developed through all of the SREM phases covered by the course. The sample system should require application of all of the methods and constructs covered in the course, and should require repeated application of the parts of SREM that are shown by the task analysis to be especially important in actually applying SREM on a project or that have been found to be especially difficult to learn. Short presentations on the material of each lesson should be immediately followed by a question and discussion session, which should be followed by instructor guided application of the lesson material by student teams. The application part of each lesson should include any relevant processing by the SREM software;

5) Each student should be provided course materials. While there may be additional materials required by the instructor for the presentation portion of some lessons, the need for these materials can only be determined during the detailed course design. The materials for this course should include:

   a) A course guide consisting of an overview and schedule as described above, a table of contents keyed to the lessons to be presented, and lesson descriptions including lesson objectives, criteria for achievement of objectives, learning activities, and evaluation procedures,
   b) A description of the sample software system to which the student will be applying SREM during the course,
   c) Enough forms and other aids of the types used in actual SREM applications to complete all of the application exercises,
   d) Reference materials describing RSL syntax, RADX

processing and error messages generated by the SREM software,

e) A text sequenced to follow the order of presentation of the lessons and including description of recommended techinques of applying SREM, examples of commonly used SREM constructs, examples and explanations of the control cards needed to execute SREM, examples of common errors made in applying SREM, and sample completed forms, drawings, translations, and REVS outputs,

f) A glossary containing definitions, with examples, of SREM terminology;

(6) A possible means of implementing the redesigned SREM course is through Computer Aided Instruction (CAI), e.g., the Unix "learn" program or the PLATO system. If a fully interactive SREM were available, it could be integrated into a program that generates lessons to teach SREM. This would provide flexibility in maintaining the course as well as providing a quicker introduction to the actual use of SREM.

5.4.2 Outline For Course Redesign

If at all possible the SREM user training course should be completely redesigned. The recommended methodology emphasizes criterion referenced instruction techniques as developed by Robert F. Mager [MAGE67,MAGE62]. The following steps are further detailed in Mager's Developing Vocational Instruction.

(1) Course Purpose

Before an improved SREM course can be designed, the overall

purpose of the course must be established. We will assume that the purpose of the course will be to train students who have the appropriate prerequisites to apply SREM in an actual work situation as described in the job description. A distinction should be made between this statement of purpose, emphasizing training and job performance, and statements of purpose emphasizing understanding or acquisition of knowledge. Because our emphasis is on preparing students for the actual application of SREM, the course will be based on a description of the job the students will be expected to perform after completing the course.

(2) Job Description

The first step in formal course development is the statement of a job description. This description consists of one or two paragraphs mentioning each of the types of things done on the job such as, in the case of SREM, drawing R_NETs and SUBNETs from functional requirements specifications. The goal at this point is completeness of the description without reference to the details of the tasks mentioned.

Input to the job description comes from interviews and observations of those already doing the job, information from documents describing the job, and the stated expectations of those who will be managing the job. When created, the job description serves as the starting point for doing the task analysis phase of course development.

(3) Task Analysis

The first step of task analysis is to list every task associated with doing the job. Most of the information required to do this

initial step will have been gathered during creation of the job description.

By consultation with managers of projects using SREM, observation of others using SREM, and consideration of our own experience in applying SREM, the tasks will be weighted according to frequency of performance, importance, and learning difficulty.

Each task will then be further broken down into detailed subtasks. For example, within the SREM task "correct RSL syntax" might be, among other subtasks, "determine validity of error messages" and "identify error messages". Each subtask will then be analysed to determine the type of performance involved (i.e., recall, discrimination or problem solving) and the difficulty of learning each subtask will be estimated.

(4) Target Population

One of the most important phases of creating a SREM user training course will be an analysis of the target population for the course. The existing skills, level of experience, education and motivations of the students for whom the course is designed must be determined, with both averages and ranges established for each student quality that is evaluated.

(5) Course Objectives

With the above steps completed, the objectives for the course will be determined. The main purpose of the objectives is to describe, with reference to measurable criteria, the kinds of student performance expected on completion of the course.

The course objectives will not cover every subtask derived in the task analysis. The main criteria for inclusion of performance of a task in the course objectives is importance to the overall job performance. Another consideration is the probability of success in learning the task in the course versus learning it on the job once other skills have been learned in class.

Each course objective will be described in terms of a goal, an associated student behavior, conditions under which the behavior will be displayed, and a criterion or criteria by which successful performance will be defined. For example, a SREM course objective might read:

Goal:	Be familiar with symbols used in drawing R_NETs and SUBNETs.
Behavior:	Match symbols with their definitions.
Condition:	Given list of symbols and definitions.
Criterion:	Match 5 out of 7 correctly.

(6) Course Prerequisites

Course prerequisites are descriptions of restrictions to be placed on incoming students. The prerequisites will be testable skills thought to be present in the target population that are applicable to performance of behaviors described in the course objectives. The inclusion of a prerequisite results in an associated course objective not being taught in the course. Entry testing for prerequisites must be done to ensure that either students not meeting these prerequisites are prevented from taking the course, remedial instruction is provided to give the students the required skills, or, if a large part of the target population does not have the required skills, the course

is expanded to include teaching those skills. The unique nature of SREM already indicates potential users need to have extensive software engineering experience before taking the course.

(7) Instructional Procedures

Instructional procedures will be chosen according to course objectives. Techniques will emphasize student performance of the behaviors described in the course objectives under conditions approximating, to whatever degree practical, the conditions under which the students will be expected to perform on the job. For a SREM course, techniques involving a team approach to applying SREM, and practice performing the various steps and phases of SREM using source documents similar to those to which they will be expected to apply SREM, will be emphasized.

(8) Sequencing Instruction

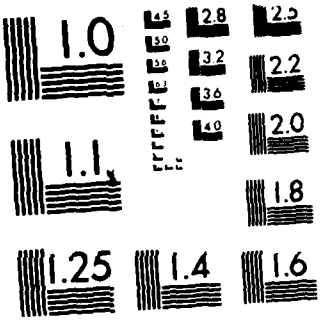
The course will be divided into a series of instructional units or lessons, each with its own objectives and performance evaluation. After the initial overview lesson, the nature of SREM will probably require that the lessons be sequenced in the order in which the behaviors taught are actually used when applying SREM.

(9) Evaluation of Student Learning

Each lesson will be followed by an evaluation procedure in which the students will be required to exhibit the behaviors described in the lesson objectives. Because of the length of time required to perform an analysis of a document using SREM, it may not be practical to have a comprehensive final evaluation to test

AD-A141 631 SREM (SOFTWARE REQUIREMENTS ENGINEERING METHODOLOGY) 2/2
EVALUATION VOLUME 1(U) MARTIN MARIETTA DENVER AEROSPACE
CO A STONE ET AL. FEB 84 MCR-83-553-VOL-1
UNCLASSIFIED RADC-TR-83-314-VOL-1 F30602-80-C-0272 F/G 9/2 NL

END
DATE FILMED
7-84
DTIC



achievement of all the course objectives.

(10) Course Materials

Course materials, aside from a course guide describing the course objectives and a schedule of lessons, will be included based on relevance to achieving the course objectives. For a SREM course, they might include: a text emphasizing the main steps of using SREM, with sample common problems and their solutions; a concise reference manual of RSL syntax, REVS control cards, and error messages; a glossary of SREM terminology, with examples; forms and other recommended aids for performing the applications exercises; and audio-visual materials if indicated by the analysis and choice of instructional materials.

5.4.3 Summary

In summary, we most highly recommend that the course be redesigned. This includes thorough task and audience analysis leading to a course with criterion-referenced objectives, application exercises and evaluation activities. Even if the course is not redesigned, its effectiveness can be significantly improved by the implementation of any or all of (1) a thorough course overview provided at the first session, (2) a subject matter overview including history, the problems SREM addresses and major functions to be studied, (3) short, clearly defined lessons with definite objectives and application exercises, (4) the inclusion of one sample system to be developed throughout the course by application exercises, (5) well-indexed and cross-referenced materials including a course guide, a glossary of SREM terminology, reference materials and a text sequenced in lesson order, and (6) possible addition of Computer-Aided Instruction (CAI).

6.0 REFERENCES

[ALFO79] Alford, M.W. et al, Requirements Development Using SREM Technology, Volume II, TRW Inc., 1979. (NADC Edition)

[ALFO80] Alford, M.W., Software Requirements Engineering Methodology at the Age of Four, COMPSAC 80 Proceedings, P. 366-374.

[ASE80] Lubbes et al, The Advanced Sensor Exploitation (ASE) Analysis/Design Report, Pattern Analysis and Recognition Corporation, 1980.

[CSID80] ----, Interpreter Software Requirements Document, Harris Corporation, 1980.

[GUNT79] Gunther, L.J. et al, REVS Users Manual, TRW Inc., 1979.

[HOPCR69] Hopcroft, J.E. and Ullman, J.D., Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.

[LOSH79] Loshbough, R.P. et al, REVS Quick Reference User's Manual, TRW Inc., 1979.

[MAGE62] Mager, R.F., Preparing Instructional Objectives, Palo Alto, California, Ferson Publishers, Inc., 1962.

[MAGE67] Mager, R.F. and Beach, I.R., Jr., Developing Vocational Instruction, Belmont, California, Fearson Publishers, Inc., 1967.

[MMSE79] ----,Software Engineering Standards, Martin Marietta Denver Aerospace, 1979.

[MYER76] Myers, G.J., Software Reliability Principles and Practices, John Wiley & Sons, 1976.

[NELS78] Nelson, R. "Software Data Collection and Analysis" (Draft-Partial Report), RADC/COEE, Griffiss AFB, N.Y. 13441; Sept 1978.

[PASI80] Pasini, C., Agneesens, H., Software Requirements Engineering Methodology (SREM) Management Overview, TRW Inc. (730 pages, structured for a 10-day course.)

[ROSS75] Ross, D.T., Goodenough, J.B., and Irvine, C., "Software Engineering: Processes, Principles, and Goals," Computer, May 1975, pp. 62-72.

[TAUS80] Tausworthe, R.J., "Software Cost/Resource Modeling", Proceedings, Fifth Annual Software Engineering Workshop, NASA/GAFC, November 1980. (See also other reports in this series. The NASA-SEL data is recorded in the RADC "DACS" software productivity database.)

[TRW79] Alford, M. et al, Requirements Development Using SREM Technology, Management Overview - NADC Edition, Volume I., TRW, Inc., June 1979. (183 pages, 60 view graphs.)

[WARB83] Warburton, R.D.H., "Managing and Predicting the Costs of Real-Time Software", IEEE Transactions on Software Engineering, Vol SE-9, No. 5, September 1983, p. 562 (Productivity formula from Putnam, p. 568.)

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

END

DATE
FILMED

7-80

DTIC