# SOFTWARE REQUIREMENTS ENGINEERING METHODOLOGY (SREM) AT THE AGE OF TWO

M. W. Alford

TRW Defense and Space Systems Group
Huntsville, Alabama

## 1.0 INTRODUCTION

The Software Requirements Engineering Methodology (SREM) was presented to the Software Engineering community two years ago at the Second International Software Engineering Conference [1]; the SREM support software, the Requirements Engineering and Validation System (REVS), was also presented then [2]. SREM was developed for the Ballistic Missile Defense Advanced Technology Center (BMDATC) to address the generation and validation of software requirements for Ballistic Missile Defense Weapons Systems -- the motivation and environment for this research has been previously described [3]. At that time, REVS was operational only on the Texas Instruments Advanced Scientific Computer (TI ASC), and the methodology had been applied to one moderate sized "proof of principle" demonstration problem.

Since then, SREM has been successfully applied to both the generation and independent validation of software requirements for several systems. REVS has been transported to a number of other host computers, and its performance has been improved. The methodology has been successfully transferred to a number of other organizations, and applied to a wider class of problems. The purpose of this paper is to provide a status report of the SREM requirements development procedures, requirements language, support software, and transfer of this technology to other organizations, and to provide an overview of plans for extensions and improvements.

Section 2.0 contains a summary of the status of REVS installations and the transfer of the SREM technology to other organizations. Section 3.0 contains an overview of the results of several diverse SREM applications. Section 4.0 contains an overview of planned extensions to SREM and REVS. Section 5.0 contains some conclusions of our experiences to date.

The objectives of the research leading to SREM were to reduce the ambiguity and errors in software requirements, to make the software requirements development process more manageable, and to provide more automation in validating the software requirements. More details can be found in references [1], [2], and [3].

The SREM requirements development procedures identify the steps and objective completion criteria necessary to define software requirements using the Requirements Statement Language (RSL) and the Requirements Engineering and Validation System (REVS). SREM thus provides a road map of the sequence of activities necessary for the definition of software requirements and the manner in which REVS can be used to ensure that an activity is complete -- thereby providing a high degree of management visibility of the requirements development process.

SREM is based on a Graph Model of Software Requirements [4] which is an extension of the Graph Model of Computation [5]. The basic concept underlying SREM is that design-free functional software requirements should specify the required processing in terms of all possible responses (and the conditions for each type of response) to each input message across each interface. Thus, functional requirements identify the required stimulus/response relationships, and autonomously generated outputs. These required actions of the software are expressible in terms of Requirements Networks (or R-Nets) of processing steps. Each processing step is defined in terms of input data, output data, and the associated transformation. Figure 1 presents an R-Net for a Hospital Patient Monitoring System [6] which accepts a measurement of the blood pressure, temperature, skin resistance, etc., for a patient, tests it for validity, records it, requests the next measurement, and tests the measurement against a pre-specified set of upper and lower limits. Note that five paths of processing are identified, which combine into three possible stimulus/response requirements -- the paths to request the next measurement and record the current measurement occur regardless of whether the measurement violates the constraints.

The concepts of the Graph Model of Software Requirements are embodied in the design of the Requirements Statement Language (RSL), a machine-processible language for the unambiguous statement of software requirements. RSL is composed of elements (e.g., R-Nets, Processing Steps, Data Messages, Input Interfaces -- the "nouns" of the language), their attributes (e.g., units of data, required response times of processing paths, descriptions -- the "adjectives" of the language), their relationships (e.g., data is input to a processing step, a message passes an output interface -- the "verbs" of the language), and structures (used to define the conditions and sequences of processing steps which comprise the required



CH1338-3/78/0000-0332$00.75 

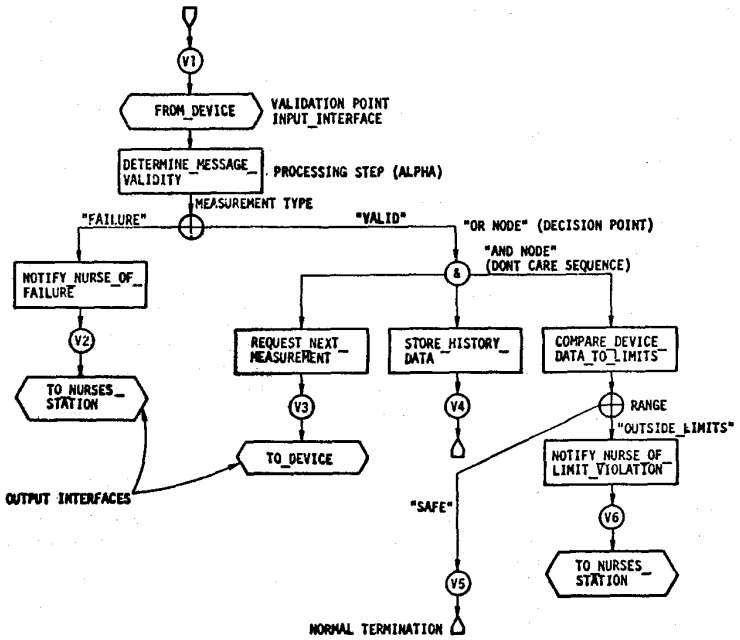


Figure 1 Example R-Net

stimulus/response relationships to be satisfied by the software). Table 1 presents a subset of the requirements for the Patient Monitoring System expressed in RSL. In addition to nouns, verbs, and adjectives for stating requirements, RSL also contains elements, attributes, and relationships that express management concepts (e.g., traceability, completeness, authorship, and version). In all, RSL is composed of 21 types of elements, 21 types of attributes, 23 types of relationships, and three types of structures. It is the structures (R-Nets, Subnets, and Paths) and their formal mathematical foundations, and the stimulus-response orientation which distinguish RSL from the traditional techniques for stating software requirements (e.g., the PSL [7] approach, or standard DoD Military Specifications [8]).

Table 1 Example RSL Definitions

```
MESSAGE:  DEVICE_REPORT.
    PASSED THROUGH:  INPUT_INTERFACE FROM_DEVICE.
    MADE BY:  DATA DEVICE_NUMBER
              DATA TYPE_MESSAGE
              DATA DEVICE_DATA.
DATA:  DEVICE_DATA.
    INCLUDES: DATA PULSE
              DATA TEMPERATURE
              DATA BLOOD_PRESSURE
              DATA SKIN_RESISTANCE.
FILE:  FACTOR_HISTORY.
    CONTAINS: DATA MEASUREMENT_TIME
              DATA HPULSE
              DATA HTEMPERATURE
              DATA HBLOOD_PRESSURE
              DATA HSKIN_RESISTANCE.
    TRACED FROM:  SENTENCE_2
                  SENTENCE_3.
    ASSOCIATED WITH:  ENTITY PATIENT.
ALPHA:  STORE_HISTORY_DATA.
    INPUTS:  DEVICE_DATA.
    OUTPUTS: FACTOR_HISTORY.
    DESCRIPTION:  "THE DATA PROCESSOR SHALL RECORD EACH VALID
                   MEASUREMENT FOR EACH PATIENT".
VALIDATION_PATH:  MEASUREMENT_OUT_OF_LIMITS.
    PATH:  VALIDATION POINT V1, VALIDATION POINT V6.
MAXIMUM TIME:  1.
    UNITS:  SECONDS.
```

REVS is a large software tool that handles a potentially large data base of requirements, therefore requiring a host computer with a large effective memory space and a moderately fast instruction rate.

REVS accepts RSL as input, translates it into an automated requirements data base, and provides a set of capabilities for analyzing and manipulating this data base. Specific capabilities include the following:

- Translation of an RSL expression of requirements into a central requirements data base.
- Extraction, under user control, of information from the requirements data base for analysis and documentation.
- Identification, under user control, of subsets of the data base for automatic consistency, completeness, and traceability analyses.
- Automated checking of the requirements data base for specific properties of data flow consistency (made possible only because of the underlying formal foundations).
- Graphical representation of the requirements structures, both on-line and off-line.

- Automated generation and execution of simulations directly traceable to the requirements definition.

The REVS program itself consists of over 40,000 executable PASCAL statements, making it the largest PASCAL program known to us. (In comparison, the PASCAL compiler consists of approximately 6,000 PASCAL statements.) An additional 10,000 FORTRAN statements perform data base management functions. The RSL Translator is produced from the Backus-Normal Form (BNF) definition of RSL using the Lecarme-Bochman Compiler Writing System from the University of Montreal, plus additional code for the definition of the semantic actions.

The SREM requirements development procedures identify in detail how RSL and REVS are used to generate and validate the processing requirements. The approach is to define functional requirements in terms of paths of processing, then to attach performance requirements to the paths.

First, all interfaces of the data processor are identified, together with the messages that cross interfaces **and the message contents. Next,** R-Nets are developed to specify stimulus/response relationships required of the software. This top-level information is then checked to assure that each output message is produced at least once. If this is not true, the processing definition is incomplete. The methodology continues by identifying data to be maintained by the software, the data flow between processing elements, and a series of consistency and completeness checks that identify errors and holes in the specification. After these functional requirements are identified and checked for static consistency, functional simulations are generated to verify the dynamic consistency of the requirements, and to derive required processing rates for interactive software systems. When all processing paths have been identified, performance requirements are derived and expressed with respect to these functional requirements. The functional requirements represent the "what" of the processing, while the performance requirements represent the "how well" attributes. Analytical level simulations can then be used to demonstrate the non-real-time feasibility of the processing requirements, i.e., that algorithms exist which can meet the processing accuracy requirements -- only a real-time software design will demonstrate the real-time feasibility of the processing requirements.

At the time of our presentations two years ago, SREM, RSL, and REVS had been applied to one moderately sized BMD problem to demonstrate capability and to illustrate the sequence of steps and associated outputs of the methodology. At that time, it was concluded that the research objectives had been achieved, but that SREM's applicability to other environments and utility in realistic software development environments had yet to be determined.

The remainder of this paper presents an overview of where we are today and where we are going in the future -- in order to definitize and automate a comprehensive software requirements methodology.

## 2.0 WHERE WE ARE TODAY

The availability of SREM technology for use on software requirements development projects is dependent on two factors: the availability of people who are knowledgeable in the SREM concepts, techniques, and procedures; and the availability of REVS to support the use of SREM. The substantial progress which has been made in both of these areas in the past two years is discussed below.

### 2.1 TECHNOLOGY TRANSFER

Technology development is fruitless without the mechanism for its transfer to others working in the field. Mere existence of tools, or codification of experience, does not necessarily lead to a transfer of working knowledge which can be exercised by others in operational environments. The true test of a methodology is whether it can be absorbed and applied by others.

To successfully transfer SREM technology to others, three techniques have been used: **transfer of documentation of the SREM procedures,** language, and software capabilities; presentation of a "short course" in Requirements Engineering; **and on-the-job training. Table 2 presents a list of the organizations which are applying these techniques. More details of the transfer are provided below.**

**Table 2 Technology Transfer**

| ORGANIZATION | TRANSFER METHOD: DOCUMENTATION ONLY | ON-THE-JOB TRAINING | SHORT COURSE | EXPERIMENTAL USAGE | APPLICATIONS: VALIDATION | SPECIFICATION GENERATION |
|---|---|---|---|---|---|---|
| APPLIED PHYSICS LABORATORY | | X | | X | X | X |
| HUGHES | | | X | X | | |
| McDONNELL DOUGLAS ASTRONAUTICS CORP. | | | X | X | | X |
| RCA | | | X | X | | X |
| TELEDYNE BROWN ENGINEERING | | | X | X | | |
| TRW | X | | X | X | X | X |
| UNIVERSITY OF CALIFORNIA, BERKELEY | X | | | X | | X |
| U.S. ARMY COMPUTER SYSTEMS COMMAND | | | X | X | | |

Some individuals at the University of California at Berkeley, and at TRW, relied only on the published documentation and achieved mixed results in later applications. Requirements engineers from Johns Hopkins University/Applied Physics **Laboratory (APL) and TRW have been provided** with the basic documentation and with some on-the-job training consisting of assistance in SREM on a problem of their choice. The on-the-job training provided an intensive information transfer, and the opportunity to correct any misconceptions about the software requirements engineering process and misunderstandings of the language concepts. For example, requirements engineers with previous experience writing software requirements usually have an overwhelming urge to try to define

a queueing scheme for buffering input messages in RSL; elimination of this design leads to statements of response time requirements which allows software designers to decide which buffering scheme meets such requirements.

In November 1977, a three-week course was held at McDonnell Douglas Astronautics Corporation (MDAC) in Huntington Beach, Cal., sponsored by BMDATC, to assist the technology transfer to a wider audience. Participants included requirements engineers from MDAC, Hughes, RCA, Teledyne Brown Engineering, and the U.S. Army Computer Systems Command. Course notes were provided to detail the methodology and to provide examples of inputs and outputs of each methodology step. All participants had the opportunity to define functional requirements for two example problems: a common example for the whole class, and then independent team projects. The course was evaluated by all as successful. Table 2 indicates partial results of this transfer: APL, MDAC, RCA, and TRW all have performed, or are performing, demonstration and/or operational projects for generating and/or validating software requirements using the technology.

Although continual on-the-job training is the ideal, the training course approach was found to be an effective and cost-effective mechanism for technology transfer. Several future courses are under consideration.

## 2.2 REVS AVAILABILITY

A critical element in the availability of SREM is the availability of REVS to support its application. Two years ago all REVS capabilities were operational, but REVS was available only on the TI ASC in Huntsville, Alabama, which had no remote access capability. This severely limited the scope of its application. Since then, REVS has been installed in several more locations on both TI ASC and CDC host computers with remote access capability. Table 3 summarizes the installation history of REVS; more details are provided below.

**Table 3 Current REVS Installations**

| LOCATION | HOST | I/O CAPABILITIES | | | |
|---|---|---|---|---|---|
| | | ON-LINE GRAPHICS | REMOTE BATCH | TELETYPE | OFF-LINE GRAPHICS |
| ADVANCED RESEARCH CENTER, HUNTSVILLE, ALABAMA | CDC 7600 | X | X | | X |
| NAVAL RESEARCH LABS, WASHINGTON, D.C. | TI-ASC | | | X | X |
| MDAC, HUNTINGTON BEACH, CALIFORNIA | CDC 7700 | | X | | X |
| TRW, REDONDO BEACH, CALIFORNIA | CDC CYBER 74/174 TSS | | | X | X |

In early 1977, REVS was transported to the TI ASC at the Naval Research Laboratories (NRL) in Washington, D.C., for experimental use by APL. The NRL version of REVS is run in an off-line batch or on-line teletype access mode, and accessed via teletype for data extraction, correction, and consistency checking. When the TI ASC in Huntsville was phased out in 1978, this became the only operational TI ASC version of REVS.

In July 1977, REVS was converted to a Huntsville CDC 7600, with access by on-line graphics and remote batch. This provided a substantial improvement in availability for demonstration projects. (As a by-product of this installation, the CDC 7600 was provided with a production level PASCAL Compiler with all defined PASCAL capabilities, and the ability to compile very large PASCAL programs with FORTRAN subroutines.) This version of REVS was partially optimized for the CDC 7600 memory structure, resulting in program execution times compatible with the development of software requirements in an operational environment.

REVS was then transported to the CDC 7600 at MDAC in Huntington Beach California, for experimental use in writing specifications for Army BMD software. Due to the lack of on-line graphics hardware, only batch and remote batch capability is supported at that installation.

REVS also was transported to the CDC CYBER 74/174 Time Share System at TRW, Redondo Beach, Calif. This version of REVS is currently being optimized to reduce execution time and on-line memory requirements to a level compatible with operational usage requirements.

Thus, REVS has become available on various CDC computers. This has been instrumental in aiding technology transfer and capability assessment. It also has supported the generation and validation of software requirements in operational environments.

# 3.0 APPLICATIONS

SREM was an out-growth on research addressing the statement of software requirements for BMD systems which are fully automated (no man-machine interactions), and are real-time control and engagement oriented (engagement rules defined before software requirements definition is initiated). In addition, SREM addressed only the development of requirements for software to be executed on a single processor (no distributed processing). Although the mathematical foundations of SREM were believed to be applicable to other types of software requirements development efforts, no explicit claims were made about the efficacy of SREM for those requirements. Since the SREM approach was published, it has been applied on a variety of projects and environments. Table 4 presents characteristics of a number of these projects. Some of these projects are competitive in nature, thus sensitive details have been omitted. Results of these applications are discussed briefly below.

Projects A through E were performed at TRW under the direction of those responsible for the development of the methodology and tools. Project A involved the definition of top-level software requirements for air defense engagement control software during the conceptual design phase of the project. This project involved the definition of requirements for processing to be distributed across more than five processors, and included the definition of man/machine interactions. Conclusions from this project included the following:

- SREM was applicable to the definition of testable requirements for the distributed processing network as a whole. The statement of such requirements enables meaningful tests to

Table 4 SREM Applications

| PROJECT | PERFORMER | TYPE SYSTEM | TYPE PROJECT | CHARACTERISTICS | | | ESTIMATED SOFTWARE SIZE (INSTRUCTIONS) |
|---|---|---|---|---|---|---|---|
| | | | | REAL-TIME | DISTRIBUTED PROCESSING | MAN/MACHINE | |
| A | TRW | AIR DEFENSE | CD PHASE RQMTS | X | X | X | 200 K |
| B | TRW | OPERATING SYSTEM | RQMTS RE-DEFINITION | | | | 20 K |
| C | TRW | REAL-TIME INFO | RQMTS VALIDATION & RE-DEFINITION | | | X | 200 K |
| D | TRW | AIR DEFENSE | DEMO | X | X | | 16 K |
| E | TRW | COMMAND/CONTROL | DEMO/RQMTS RE-DEFINITION | X | X | X | -- |
| F | TRW | REAL-TIME INFO | CD PHASE RQMTS | | X | X | 150 K |
| G | TRW | RADAR CONTROL | CD DEFINE RQMTS | X | | | 100 K |
| H | RCA | TEST SOFTWARE | CD DEFINE RQMTS | | | | 100 K |
| I | APL | COMMUNICATIONS | DEMO/RQMTS. VALIDATION | X | | X | 50 K |
| J | MDAC | BMD | DEMO/DEFINE RQMTS | X | | | 200 K |
| K | UCB | REACTOR CONTROL | DEMO/DEFINE RQMTS | X | | | -- |
| L | TRW | RELATIONAL DB | DEMO | | | | N/A |
| M | TI | RELATIONAL DB | DEMO | | | | N/A |

be performed after integration of the processors into the distributed network.

- The allocation of the network processing requirements to processing nodes was supported by REVS in an awkward fashion. Several approaches to ease this transition were identified but not implemented.
- The definition of the overall processing requirements provided the framework for identification of the man/machine requirements allocation issues (i.e., what role a man can play in an engagement in terms of response times, judgement, capability, and the need to maintain positive control over the weapons).

Project B involved the definition of the requirements for an existing operating system for which software requirements were undocumented; some extensions to the operating system were contemplated, and, hence, requirements on the existing operating system capabilities had to be definitized in order to address requirements for the augmented system. Conclusions from this project included the following:

- Software requirements for an operating system can be written (contrary to popular opinion that operating systems are designed without requirements).
- A major known "bug" in their operating system (i.e., the removal of a file directory could result in making the files in that directory inaccessible to all users) was found to be a requirements problem, not a design problem.
- Requirements can be extracted from existing description and design documents; the SREM concepts for stating software requirements provide considerable aid in separating the requirements from the design.
- Methodology steps for definitizing requirements for existing software follow the same steps as those for generating software requirements, i.e., first identify the interfaces, the messages crossing the interfaces, their contents, the required stimulus/response relationships, etc.; thus, the SREM steps for using RSL and REVS were applicable to address the problem.

Project C involved the application of SREM to the redefinition of a software specification for an interactive (man/machine) near real-time information management system. A major conclusion from this project was that the redefinition of software requirements follows essentially the same steps as the generation of software requirements using SREM. The major problem in performing the validation was the identification of the required sequences of processing steps defined in the various processing "functions", and identification of the data flow between processing "functions" -- just the problems that would have to be addressed to make the requirements testable.

Project D involved the redefinition and validation of requirements for existing air defense engagement software, in order to address the inclusion of requirements for new capabilities involving distributed processing. Again, the requirements were extracted from a combination of existing requirements and design documents, and sometimes from interpretation of the code itself. Conclusions from this project included the following:

- The SRE Methodology is, with little modification, applicable to requirements redefinition. The technique of first redefining the requirements, and then modifying the requirements to assess the impact on the existing software, yields significant advantages in comparison to "patching the code one more time".
- The simulation generation capability of REVS provides a rapid, cost-effective means for the generation of both functional and analytical simulations of required processing. In particular, the consistency and completeness checking capabilities of REVS speed up the process of debugging the simulation of the processing, and produce a simulation directly traceable to the requirements.

Project E involved the redefinition of top-level man/machine processing requirements for existing distributed software in order to provide an orderly means of defining augmentations, modification, and up-grade in performance requirements. This project is still on-going. Preliminary conclusions from this project include the following:

- Definition of the requirements for a network of distributed processors in terms of the stimulus/response relationships of the network provides an effective approach for separating requirements from design.
- The SRE Methodology is, with little modification, applicable to requirements redefinition and augmentation problems.
- When these requirements are established, performance improvements for the system can be allocated to require performance improvements of the individual nodes in an objective manner. In other words, application of SREM provides the tools and viewpoint to address performance improvement in a top-down manner.

Projects F and G involved the generation of requirements for software during the conceptual development phase at TRW. Project F addressed the definition of requirements for a real-time distributed information management system, while Project G addressed requirements for software to control a radar and process its data. Both of these projects started with a customer-furnished system concept, and were to generate preliminary software requirements, a preliminary design of the software, and a fixed-price bid on the software development in about six months. Similar conclusions were arrived at in both projects:

- It is difficult to develop complete and consistent, testable software requirements on that kind of schedule with any technique, even SREM. The software requirements development process is squeezed between the system design (selecting hardware, system operating rules) and the preliminary software design (to a sufficient detail to enable a fixed-price bid). Traditionally, the specification problems are swept under a rug by writing a fuzzy software specification; SREM makes the requirements and the quality quite visible (e.g., it is entirely obvious from the automated checks of REVS whether or not all messages have been generated by the proposed functional processing, and whether or not the input/output data relationships are consistent).
- The use of SREM, particularly in the initial stages of drawing the first R-Nets, provides the communication mechanism for more meaningful discussions for clarifying the initial customer requirements. The necessity of identifying stimulus/response relationships in the R-Nets makes the top-level system logic meaningful to the user, identifies ambiguities in the initial requirements, and provides the benefits of structured walkthroughs in the very early requirements phase so the customer can identify misconceptions on the part of the requirements engineers. The customer expected perhaps a dozen questions of clarification on the requirements; to define R-Nets, he received many more questions and then commented on the R-Nets to identify misconceptions.

Project H was performed by RCA to successfully demonstrate the utility of SREM in defining requirements for test software. This comes close to the traditional batch-oriented data reduction type of software. SREM had the effect of making the requirements more clear and understandable than the traditional methods of writing requirements for such software.

Projects I, J, and K have just begun; no significant conclusions have yet been reached. They are, however, being carried out as demonstrations to provide information about the utility of SREM in operational environments.

Projects L and M took advantage of the user-extensibility features of REVS to define a new language for creating a relational data base. One significant feature of REVS is that RSL is user-extensible (i.e., new elements, attributes, and relationships can be defined by the user), and in the same (or subsequent) runs, information of this type can be translated by REVS into the automated data base, and then REVS can use these definitions to document and analyze these relationships. In Project L, Texas Instruments defined their own language to identify the type of information which might be stored to manage an on-line data base development of software. In Project M, TRW is developing an experimental on-line data base to aid in the rapid estimation of characteristics of data processing architectures to support advanced sensors. In both cases, REVS has been found to have the ability to define proposed data base contents in terms of elements, attributes, and relationships; define a language to implement it and use it experimentally, both for data input and retrieval, in order to explore the utility of the concepts. REVS thus provides a laboratory for experimenting with relational data base concepts due to the flexibility of its design.

### 3.2 EVALUATION

Only one of the above applications involved the definition of requirements for U.S. Army Ballistic Missile Defense software, the original target for SREM creation. The other projects involve SREM usage in roles not originally addressed (e.g., specification validation, specification redefinition, and augmentation), and having characteristics not originally addressed (e.g., distributed processing, man/machine interactions).

The specification validation role is a natural application for SREM, requiring very little modification of the steps, and no modification of RSL or REVS. It can be applied to a project with a preliminary software specification in order to methodically identify specification problem areas, or to a project already in software development in order to verify the completeness, consistency, and testability of the requirements. This role has been found to be a near-ideal type of project to demonstrate the utility of the SREM concepts in specific operational environments, to provide the vehicle for on-the-job training, and to provide useful results. It is one of the best mechanisms available to satisfy the new regulations (e.g., DoD Regulation 5000.29) requiring validation of software requirements before proceeding with engineering development of a large system.

The specification redefinition role is similar to the above role in that the requirements for a piece of software are defined first and then augmentations are defined with respect to the baseline specification. This approach provides significant advantages to this software augmentation process, i.e., it provides for a precise definition of the required augmentations, allows augmentations to be discussed in terms of requirements instead of specific design approaches, and provides testable requirements for the end product -- the software.

Both SREM and REVS are applicable to the definition of distributed processing and man/machine interaction software requirements as they currently exist; improvements have, however, been identified. The application of SREM to these types of software

provides a top-down viewpoint and testable requirements on the processing as a whole, in addition to detailed requirements directly traceable to these top-level requirements. It thus provides the framework for making the decisions for allocating the processing to the distributed nodes, for identifying communication requirements, and for allocating the total processing requirements between the software and the manual procedures. Experience with these problems has led to the identification of extensions and augmentations to RSL, REVS, and the SREM procedures. These will improve the ability to develop the detailed specifications from the top-level specifications which are directly traceable, and to provide automated support for the partitioning process.

## 4.0 PLANS FOR EXTENSIONS AND IMPROVEMENTS

The applications discussed in Section 3 provided experience in addressing a wider class of problems (i.e., distributed processing man/machine interactions, operating systems, on-line information systems) in various roles (i.e., specification generation, specification validation, specification redefinition and augmentation) and in various operational environments (e.g., conceptual development phase working with system and software designers vs. independent validation). Assessment of this experience has led to the identification of several types of extensions and improvements to SREM, RSL, and REVS. Specific areas of research and development currently underway include the following:

- REVS operational improvements.
- Distributed Processing Augmentations.
- Smoother transition to Process Design.
- Smoother transition from System Engineering.
- Smoother transition to Software Validation and Test Planning.
- Extensions to Business Data Processing problems.

Each of these is discussed in more detail below.

### 4.1 REVS OPERATIONAL IMPROVEMENTS

As used in an operational environment, it becomes cost-effective to transport and optimize REVS for specific installations, and to add user-oriented capabilities. TRW is optimizing the performance of REVS on the CDC CYBER 74/174 TSS installation by reducing run time for frequently used operations, and reducing memory size required to execute REVS. Although this work will address the operational characteristics of REVS on a particular installation, most benefits are to other installations. In addition, a number of transferrable features have been identified to allow the user more facility in manipulating the entire requirements data base (e.g., insert a "data entered" attribute for all data base elements quickly). Their incorporation into REVS has been planned. These improvements are of the kind expected when any utility software is placed into an operational environment (as opposed to a laboratory or research environment).

### 4.2 DISTRIBUTED PROCESSING AUGMENTATIONS

In the course of generating specifications for processing which ultimately would be distributed among several geographically distant nodes, extensions to RSL and REVS were identified which would automate the process of producing a specification for the nodes from the specification of the network, and define requirements for the communication links. In this way, the specification for the processing at a node could be automatically checked for consistency against the specification for the network processing, and modifications to the node specification could be automatically incorporated in the network specification.

A methodology for performing the allocation of distributed processing to the processing nodes, and selecting the data processing hardware configurations and software design, has been a topic of separate research. The results of this research are planned to be incorporated into SREM to form an integrated methodology for the definition of engagement-oriented distributed processing systems.

### 4.3 PROCESS DESIGN INTERFACE

One of the lessons learned in the experience of using SREM in the Concept Definition (CD) phase is the necessity of quickly closing the loop between the software requirements and the preliminary design of the software. The process design of real-time software addresses the problem of defining the software tasks (the schedulable units of software), the global data base, and task scheduler which allows satisfaction of the response time requirements at the expected loads. In the CD phase environment, changes to the requirements occur with frightening rapidity and volume, and good mechanisms are required to keep track of these changes and assure that they have been translated into the preliminary process design.

Assessment of this experience has led to the identification of techniques for integrating the preliminary design concepts into the requirements data base, thereby assuring consistency of requirements and design. Plans for performing this augmentation are underway.

### 4.4 SYSTEMS ENGINEERING INTERFACE

A similar lesson learned from using SREM in the CD phase was the necessity of quickly responding to changes in the system design. Identification of different system hardware components, or different ways of using the components to achieve the system mission, leads to new or modified requirements on the data processing to be performed. During the CD phase, these system changes can occur rapidly, and require quick identification of the impact on data processing feasibility, as well as integration into the data processing requirements. Techniques for rapidly translating system design changes into software requirements changes are under development, and the integration of these techniques into SREM is in the planning stages.

### 4.5 SOFTWARE VALIDATION INTERFACE

It was identified early in the development of SREM that the technique of defining software re-

quirements from the stimulus/response point of view would have a substantial impact on the software test planning cycle, both in terms of activities to be performed, and the techniques for planning and managing these activities. For example:

- During the software design phase, the software design can be validated to assure that all processing paths in the requirements exist in the software, and that only those paths exist (except for error detection and error handling).
- Invariants can be identified from the requirements which can be proven to be met by the software design and code.
- Software tests can be validated against simulation of the required processing before being applied to the real-time software, rather than debugging the real-time software and the test software concurrently.

The full implications of testing to a complete and consistent, testable software specification have not yet been identified, but it appears that the impact of SREM on the software validation process is substantive. Research is currently being planned to address this subject.

### 4.6 APPLICATION TO BUSINESS DATA PROCESSING

A significant difference between Business Data Processing (BDP) and BMD engagement software requirements development is that the systems analysis in BMD is performed by the BMD systems analysts, while in BDP problems, the systems analysis is performed by the software systems analysts. Thus, in BDP problems, the software requirements role shifts from one of translating systems requirements into software requirements, into the role of defining the system actions and then representing them as software requirements. This shift in roles has a substantial impact on methodology and tools necessary to support such requirements definition activities, in addition to the previously discussed attributes of man/machine interface requirements definition and identification of distributed processing requirements.

Research to adapt the concepts of SREM, and its extensions to the distributed processing, man/machine processing, and interface with systems engineering activities to the Business Data Processing environment, is currently underway.

## 5.0 CONCLUSIONS

There has not yet been time, since the availability of REVS, for a project to go from a conceptual design to a software specification written using the SREM technology, to a complete software design development, and test. However, from the current experiences of using SREM, in both demonstration and actual software requirements development activities, we can draw the following conclusions:

1) With the increasing availability of REVS on CDC computers, REVS is maturing to the level of capability necessary to support the definition of software requirements in operational environments.
2) SREM and REVS now have demonstrated utility in operational environments in defining and validating requirements for a wide class of software, with the domain of demonstrated applicability increasing. Augmentations to REVS have been identified to improve the capabilities of SREM to deal with an expanded class of problems.
3) The SREM technology has been successfully transferred to others for specification generation, and specification validation activities.

Thus, it appears that SREM, even at the age of two, successfully addresses the problems of defining and validating software requirements.

## 6.0 REFERENCES


1. M. W. Alford, "A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, Jan. 1977, pp. 60-69.
2. T. E. Bell, D. C. Bixler, and M. E. Dyer, "An Extendable Approach to Computer-Aided Software Requirements Engineering", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, Jan. 1977, pp. 49-60.
3. C. G. Davis and C. R. Vick, "The Software Development System", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, Jan. 1977, pp. 69-84.
4. M. W. Alford and I. F. Burns, "An Approach to Stating Real-Time Processing Requirements", presented at Conf. on Petri Nets and Related Methods, Massachusetts Institute of Technology, Cambridge, MA, July 1-3, 1975.
5. V. C. Cerf, "Multi-Processors, Semaphores, and a Graph Model of Computation", Dept. of Computer Science, University of California, Los Angeles, CA, Report UCLA-ENG-7223, April 1972.
6. W. P. Stevens, G. F. Myers, and L. C. Constantine, "Structured Design", IBM Systems Journal, Vol. 13, No. 2, 1974, pp. 115-139.
7. D. Teichroew, E. Hershey, and M. Bastarache, "An Introduction to PSL/PSA", ISDOS Working Paper 86, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, March, 1974.
8. Department of Defense, "Military Standard Specification Practices", Report MIL-STD-490, October 1968.