Requirements definition and its interface to the SARA design methodology for computer-based systems

by JAMES W. WINCHESTER
Hughes Aircraft Company
Fullerton, California
and
GERALD ESTRIN
University of California
Los Angeles, California

ABSTRACT

This paper presents results of efforts during 1979–1981 to integrate and enhance the work of the System ARchitects Apprentice (SARA) Project at UCLA and the Information System Design Optimization System (ISDOS) Project at the University of Michigan. While expressing a need for a requirements definition subsystem, SARA had no appropriate requirements definition language, no defined set of requirements analysis techniques or tools, and no procedures to form a more cohesive methodology for linking computer system requirements to the ensuing design. Research has been performed to fill this requirements subsystem gap, using concepts derived from the ISDOS project as a basis for departure.

INTRODUCTION

Research into requirements definition and design methodologies for Computer-based Information Processing Systems (CIPS) has been extensive. Some fundamental concepts have emerged:

1. *Hierarchical decomposition* from abstract descriptions to refined detail[1]
2. *Verification* analysis to ensure that each level of description is consistent with and traceable to adjacent levels[2]
3. *Simulation* as a legitimate analysis aid in detecting incomplete and inappropriate designs[3,4]

Many methodologies utilize notations that ease the burden of analysis and decomposition as well as provide a vehicle that enhances understanding and design freedom. Of principal interest are *pictorial* and *graphical modelling,*[5] specialized *textual* languages,[6,7,8] and *database* management of information.[9]

Less prevalent before the SARA research[10–13] was recognition of the following needs: (1) describing *attribute* requirements along with *process* and *function* requirements, (2) modelling CIPS *structure* as well as *behavior,* (3) separating models of the *environment* and the CIPS and modelling the environment along with the CIPS, and (4) constructing *tests* for requirements satisfaction as a necessary adjunct to defining the requirements. Nearly all methodologies concentrate separately on either the requirements phase or the design phase of the CIPS development cycle. That narrow concentration creates an artificial gap in notation and analysis between the requirement and design phases, generally resulting in ad hoc methods to bridge this gap. More coherence between the requirements definition and design methodologies is needed, not only to bridge this gap but to close or eliminate it.

In this paper, the authors discuss the results of efforts to relate requirements definition and design methodologies by integrating and enhancing the work of the System ARchitects Apprentice (SARA) Project at UCLA and the Information System Design Optimization System (ISDOS) Project at the University of Michigan. SARA[12] offered support to a designer in creation and analysis of multilevel models. While expressing a need for a requirements definition subsystem, SARA had no appropriate requirements definition language, no defined set of requirements analysis techniques or tools, and no procedures to form a more cohesive methodology for linking requirements to the ensuing design. The ISDOS Project's PSL/PSA System[9] offered support to problem statements, problem analysis, and management of resulting information but had no other support to give to the design process. Research has been performed to fill the SARA requirements subsystem gap, using concepts derived from the ISDOS project as a basis for departure. Neither the PSL/PSA nor the SARA systems were looked on as models of perfection in supporting computer-aided creation of complex systems whose behavior would satisfy customers' and designers' intents. They each offered some unique strengths but also needed each other's strengths.

This paper is organized as follows. The system development life cycle, specifications and requirements categories, and analysis aspects of requirements definition are first summarized.

The SARA methodology and the PSL/PSA system are then briefly described. The framework of SARA, augmented with a requirements definition subsystem derived from the concepts of PSL/PSA, is proposed as a viable approach to an integrated development methodology.

The requirements definition subsystem can be characterized by three components: (1) a *Requirements Definition Language (RDL),* (2) *Requirements Analysis Techniques and Tools,* and (3) *Requirements Definition Procedures.* Due to the limitations on publication length, the emphasis of this paper is on the RDL and its semantic foundation. Details of the complete requirements definition subsystem and its interface with the SARA methodology are found elsewhere.[13]

Finally, a brief summary of experience using RDL and the state of the support tool development is described.

OVERVIEW OF REQUIREMENTS DEFINITION ISSUES

The requirements for a CIPS must be recorded in some fashion to provide a means of communication between individuals and supporting tools involved in its design. This record consists of a set of specifications comprising language statements (natural or some special language), graphs, diagrams, and tables. Determination of the form and format of these specifications is an important issue. Its resolution is affected by the desired interface between the requirements specification and the succeeding design processes.

The requirements specification must include three categories of requirements. First are function requirements. Function requirements specify the transformations that a system must perform. Second are process requirements. Process requirements specify coordinated sequences of functions. Third are attribute requirements. These are statements of constraints and performance parameters imposed upon elements of the CIPS.

A means must exist to analyze the requirements specification to ensure that certain criteria for a well-formed specification are satisfied. The specification should be *understandable*

to those who are providing the requirements information (the "customers") as well as those responsible for developing the proposed system (the "designers"). The information within the specification should be *consistent*; i.e., no subset of requirements should be incompatible with any other subset. The specification should be *complete* so that unintended value judgements can be avoided during the design process. The information within the specification should be *traceable* to the resulting design and implementation to verify that the resulting CIPS has addressed all requirements. The requirements should be *testable* to validate that the resulting design satisfies all of the requirements. The requirements should be *realizable* in the sense that there are no unattainable requirements which are detectable. Finally, the requirements should be specified so that there is *design freedom* allowed wherever possible.

SARA METHODOLOGY

The SARA methodology uses a set of tools and procedures to design computer-based information processing systems. The SARA methodology has evolved from research and development continued since the early 1960's at UCLA.[11]

An overview of the SARA methodology is illustrated in Figure 1. The methodology is characterized as requirement-driven; that is, requirements that the CIPS must satisfy are specified, and the design activity proceeds to create a system that can meet those requirements. The environment with which the CIPS is assumed to interact is explicitly defined at the beginning of the design process. Validation of a CIPS' design is meaningful only in the defined environment.

To describe and evaluate CIPS (and the set of decomposed subsystems) a collection of modelling tools is used.[10,11,12,14] A *structural* model identifies subsystems and their interconnections. A set of *behavioral* models[14] expresses the behavior of subsystems and their behavioral interrelationships. The Graph Model of Behavior (GMB) consists of two separate but interrelated *control* and *data* graphs to express behavior. An interpretation model is associated with each processor and data set. A GMB model can then be exercised through an interpreter that simulates the behavior represented in the model, providing a means to evaluate ensuing CIPS designs. A control flow analyzer[15] is used to detect pathologies such as deadlock.

Once a CIPS has been partitioned to the point at which the designer feels confident in understanding each subsystem and knowing how to fabricate it, the *composition* process begins. The designer composes the subsystems using validated building block models of existing hardware, software, and other elements that may be used to enhance analysis. The specifications for the building blocks should be consistent in form with top down requirements. In the limit, if a building block exists whose specification satisfies a requirement which was generated top down, its acceptance should be simple. Research is ongoing to discover canonical forms for specification of existing hardware and software elements.[16,17] The composed models of the subsystems are then analyzed and tested using the

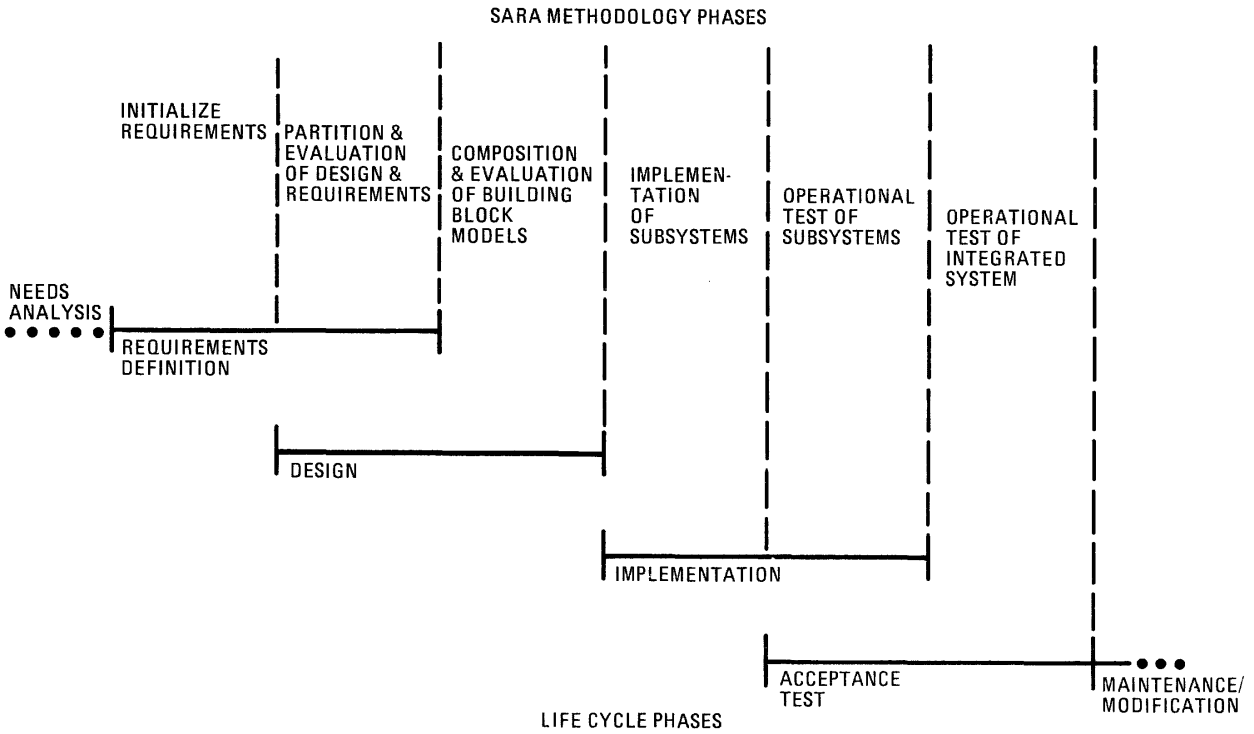

Figure 1—The CIPS development life cycle phases as constrained by the SARA methodology phases

modelling and simulation tools to verify that the subsystems and, ultimately, the complete CIPS satisfy all requirements. The physical implementation of the CIPS can then be fabricated directly from the building blocks used in the model of the CIPS.

PSL/PSA SYSTEM

The PSL/PSA system is a product of the Information System Design Optimization System (ISDOS) Project. The ISDOS Project is an ongoing research effort at the University of Michigan.

The Problem Statement Language (PSL) consists of a syntax and semantics for describing requirements according to a structured format of objects and relationships. Aspects of system structure, size, volume, dynamics, properties, data structure and derivation, and project management can be described. Included in the language is the capability to add descriptive English language comments and definitions of object attributes.[9]

The Problem Statement Analyzer (PSA) consists of all the computer software to process, analyze, and manage the PSL statements and the resulting database of PSL information. Final complete documentation of the PSL database can be produced by PSA semiautomatically in desired formats.

REQUIREMENTS DEFINITION FOR SARA

The *SARA methodology is based upon accurate specification of requirements for a CIPS being designed.* The methodology includes tools and procedural steps for decomposing, composing, and modelling CIPS to create a design that tries to meet the desired requirements and can be directly implemented. Figure 1 illustrates how the system development life cycle phases can be defined in terms of the SARA methodology phases. The requirements definition phase is seen to form the basis for all of the decomposition (refinement) activity. Thus the requirements specifications form a continuous stream of documentation of the CIPS from the most abstract customer-defined need for the CIPS down to the detailed refinement of subsystems, so that the designer can construct the subsystem from existing building blocks. In this context, the concept of distinct design specifications is not necessary; the design specifications can be a refined level of the requirements specifications.

A REQUIREMENTS DEFINITION LANGUAGE FOR SARA

The Requirements Definition Language (RDL) is used to express the requirements of a CIPS and to interface those requirements to the ensuing SARA oriented design. To determine what elements of information must be included in a requirements definition, one must have an appropriate model, or representation, of a computer-based information processing system (from requirements definition and design definition viewpoints) and an appropriate model of a requirements specification.

Computer-based Information Processing System Semantic Model (CIPSSM)

RDL's semantic model of a computer-based information processing system is based on SARA's structural and behavioral models of a system.[10,12,18,19] Most requirements for a system deal with conceptual information as opposed to physical realization. Therefore a semantic model, from a requirements viewpoint, should be concerned with conceptual constructs *onto which physical constructs can be mapped as part of the design activity*. The CIPSSM is derived from this basis.

CIPSSM primitives

The CIPSSM consists of six primitives that can be combined to model the structure and behavior of a CIPS and its environment. This representative framework allows the requirements, design, and implementation information to be associated with the model as the system development proceeds from requirements definition through implementation. Inherent in this modelling approach is the ability to perform a controlled refinement of the primitives to create a multilevel representation of a CIPS. At each refinement, more descriptive details are added to effect a progression from the conceptual requirements to physical realization.

The CIPSSM structural model primitives are *systems, dataflows,* and *connectors*. The CIPSSM behavioral model primitives are *functions, data-uses,* and *processes*. Figure 2 illustrates the graphical representation of the primitives and provides a brief description of the meaning of each primitive.

CIPSSM rules

The CIPSSM primitives interact with each other according to well-defined rules. The semantic rules are organized into three categories: (1) allowed relationships between the environment and CIPS domains of the design universe (the only allowed interface between the environment and the CIPS is through connectors, data-flows, and processes); (2) allowed relationships between primitives within each domain (e.g., a function can derive any number of data-uses); and (3) allowed decomposition relationships (e.g., a function can consist of any number of subfunctions). The RDL is designed to implement these semantic rules while the Requirements Definition Techniques and Tools are designed to enforce them.

Requirement Specification Model

The Requirement Specification Model (RSM) is a definition of the form and format of the requirements specifications for a CIPS. RDL is the principal language that will document the information that must be included in the RSM. The RSM ensures that the criteria for a well-formed specification are achievable and that the three categories of requirements are discernible. To perform this task, the RSM is set up to provide a means to describe a CIPS at all stages of its development and then automatically extract the function, process, and attribute requirements identified in the description. After the require-

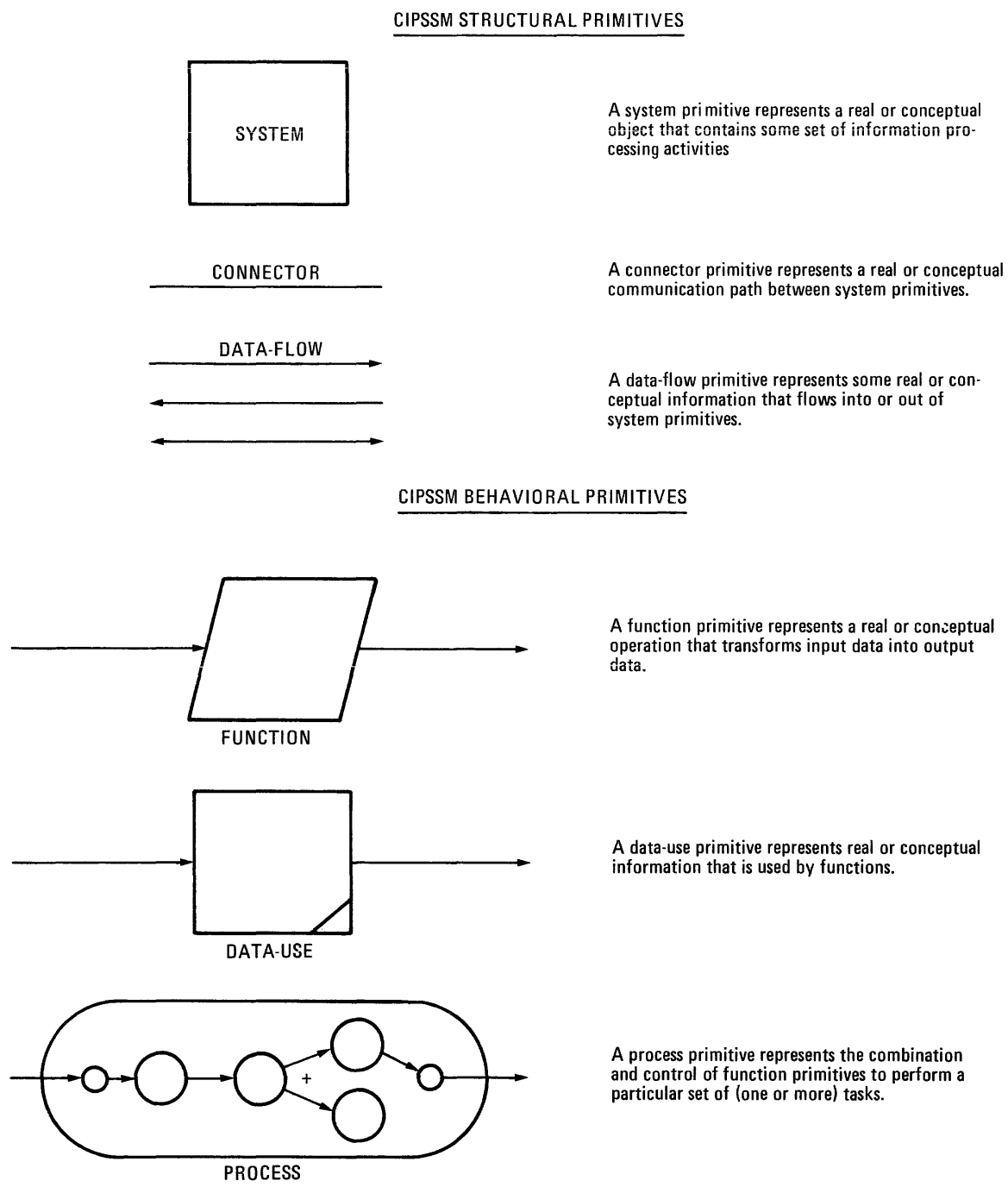

Figure 2—The graphical representation and definition of the CIPSSM primitives

ments are extracted, the tests for requirement satisfaction can be defined.

CIPS views

The RSM provides a description of four interrelated views of the CIPS. These views are decomposable in a structured fashion so that, within any particular view, the information at level (i + 1) is related to the view from level (i). The views adhere to the semantic rules of the CIPSSM. An example of the graphical representation of the four views using CIPSSM primitives is illustrated in Figure 3. The four views are designed to form a composite of the CIPS and its environment. All information expressed graphically in the four views and

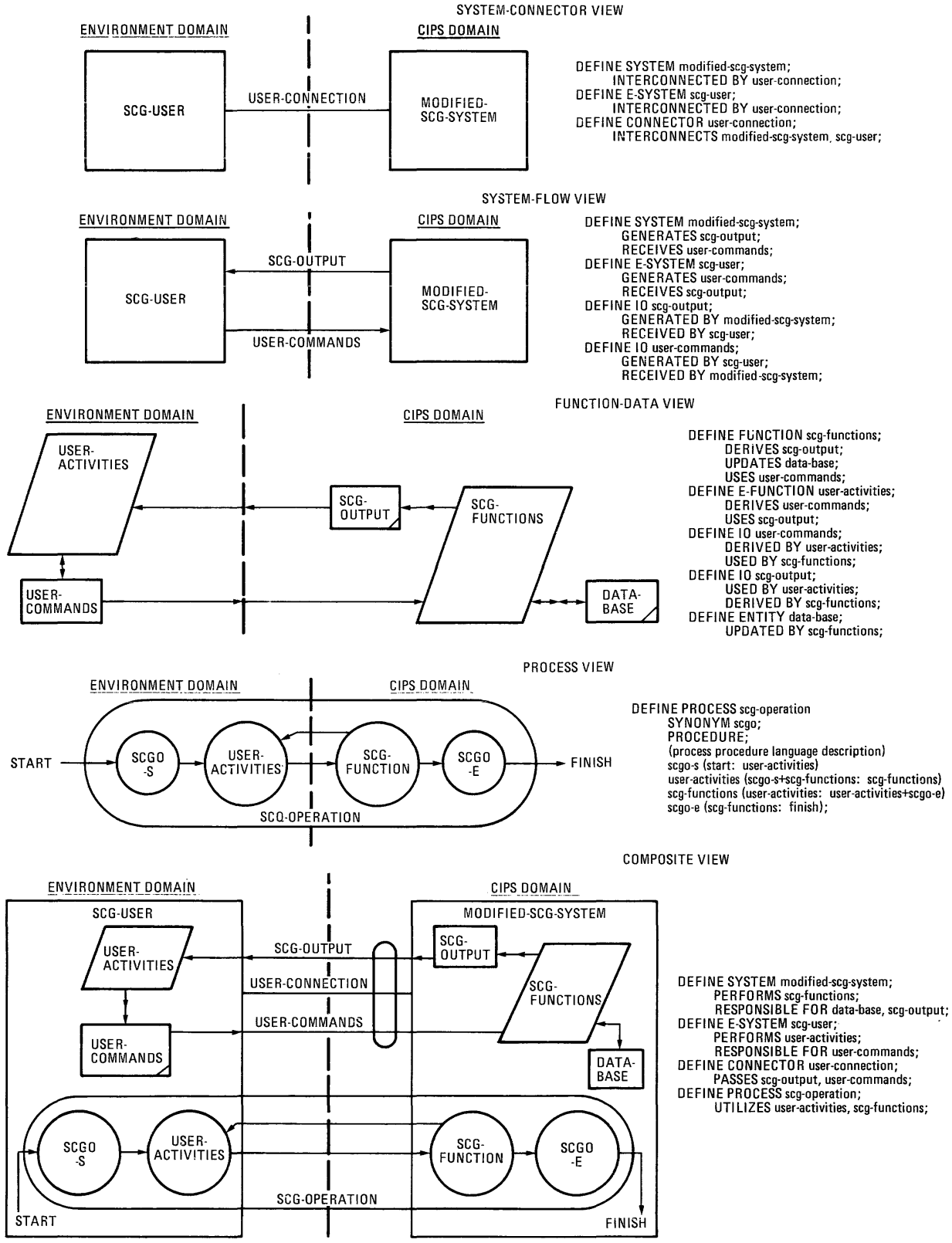

Figure 3—An example of the four CIPS views and the corresponding RDL representation for a modified structure chart graphics (SCG) system

their composite have a corresponding RDL textual representation. RDL also allows a means to express detailed information about CIPSSM primitives that cannot be represented graphically.

CIPS view decomposition

Requirements decomposition proceeds from a primarily logical description of a CIPS to a physical description of the CIPS that ultimately represents the actual design. The four descriptive views of a CIPS are oriented toward gradiated levels of logical versus physical description, between the views, and within the views.

The requirements decomposition process proceeds in parallel with the design process after the initial requirements specification is completed. The requirements decomposition is made *as the result of design decisions*. Figure 4 illustrates the graphical representation of a decomposition of the function-data view shown in Figure 3. At each level of decomposition, the designer gets a new set of requirements to respond to; however, since the new set of requirements were derived and documented from the previous level of requirements, continuous traceability is maintained between one step of decomposition and another. The RSM, built upon the CIPSSM primitives and documented by the RDL, is appropriate for the description of the CIPS at all stages of the requirements definition phase of the development cycle, as displayed in Figure 1.

Requirements extraction

Once a satisfactory *requirements specification level* is defined, using the CIPS graphical views and corresponding RDL descriptions, a complete set of function, process, and attribute requirements can be extracted. The function requirements are the RDL descriptions of the function primitives that are portrayed in the function-data view. The process requirements are the RDL descriptions of the process primitives that are portrayed in the process view. The attribute requirements are the RDL descriptions of attributes associated with all of the primitives in all four views. The requirements extraction concepts are illustrated in Figure 5. The function requirement extracted from the initial CIPS description of Figure 3 is presented in Figure 6(a).

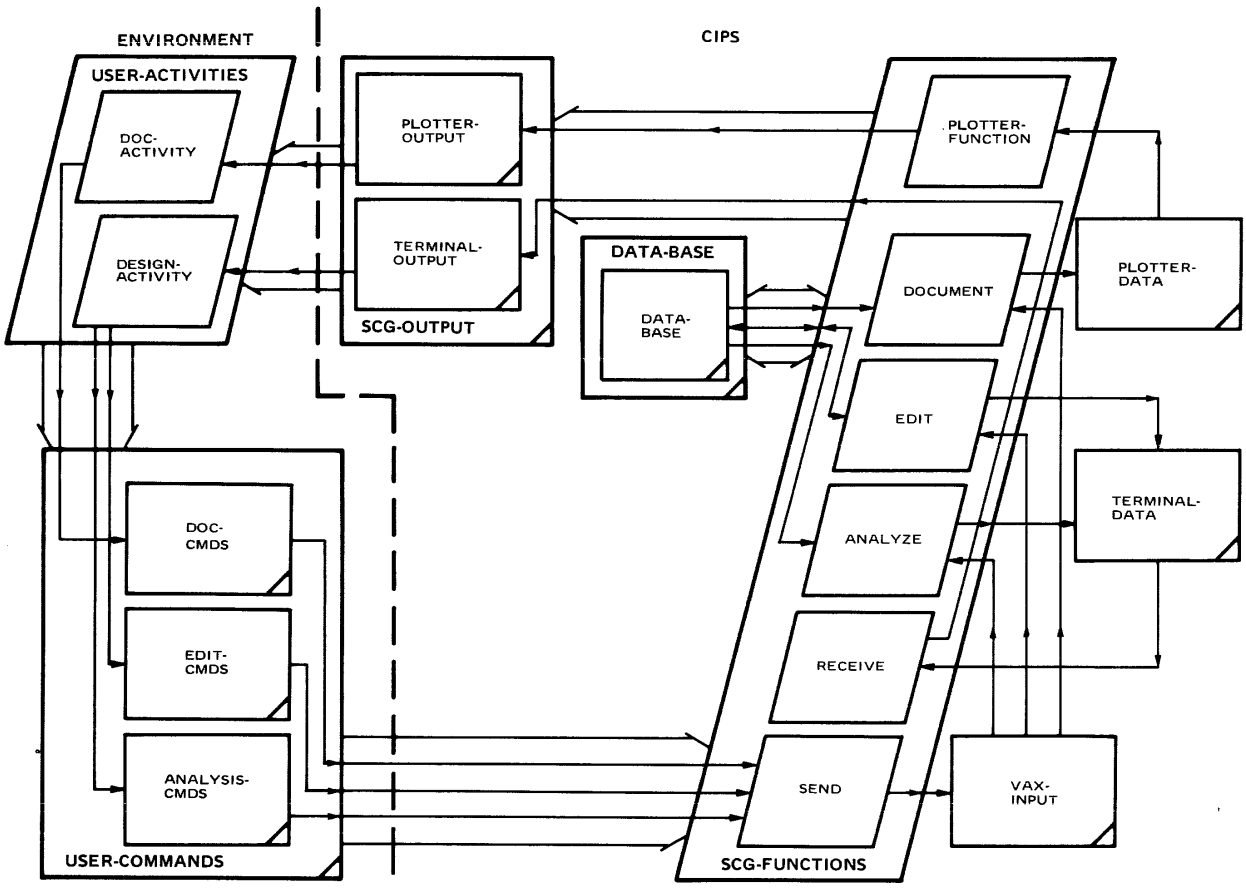

Figure 4—An example decomposition of the function-data view shown in Figure 3

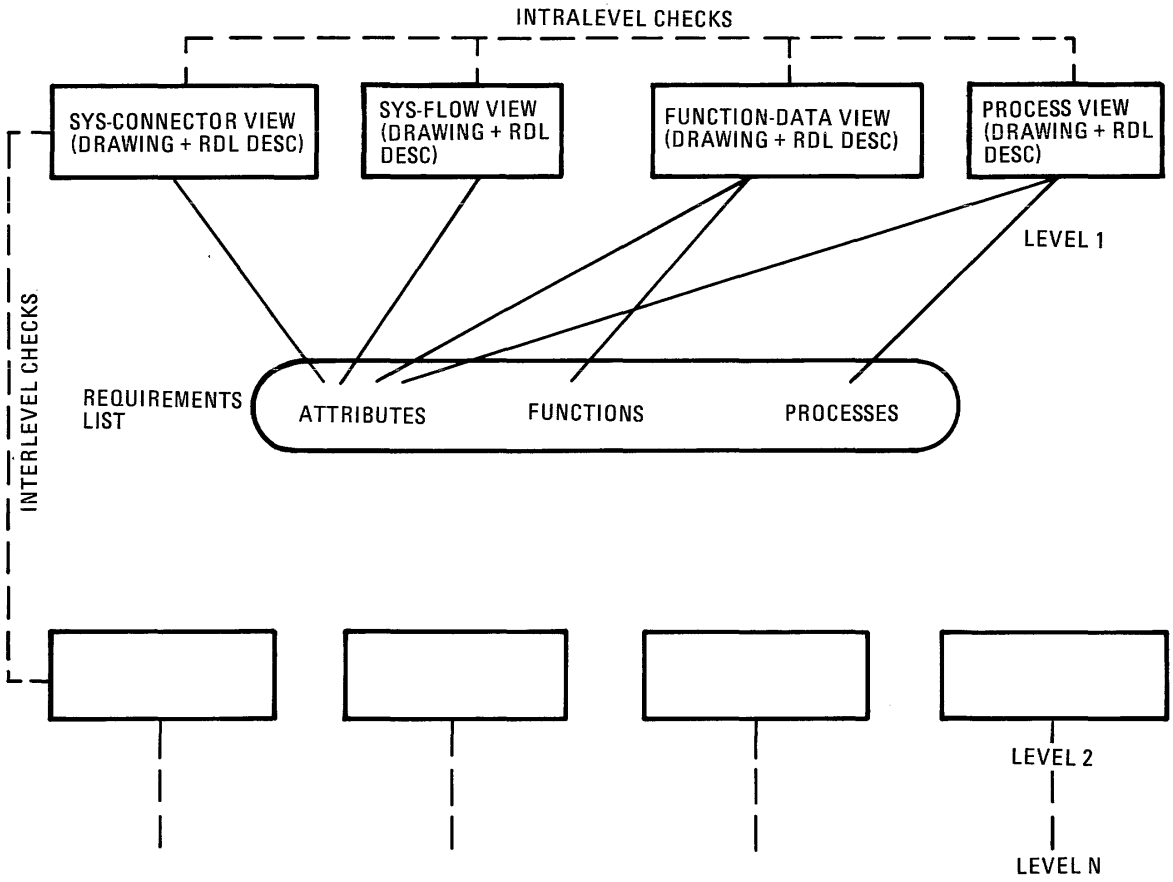

Figure 5—At every level of specification, requirements are extracted from the complete CIPS description as represented in the four views

Requirements tests

At every requirements specification level, a list of function, process, and attribute requirements is generated. At every level, each unique requirement makes one or more associated tests mandatory, so that the requirement can be verified and validated. The tests consist of test procedures and criteria to judge the outcome of the tests (i.e., whether the requirement is satisfied or not). In addition, the tests should reflect the multilevel decomposition of the requirements by becoming more detailed during refinement.

The nature of the test procedures and criteria depends upon the category of requirements being tested. RDL provides constructs permitting definition of initial conditions, final conditions, and procedures for defining how the test cases should be executed and acceptance criteria for determining how the outcome of each test case will be evaluated. A test for the requirement extracted in Figure 6(a) is described in Figure 6(b).

Requirements specification outline

The body of the requirements specification is organized by requirements specification level. Each level of the document consists of the four drawings of the CIPS views; a composite drawing of the four views; the function, process, and attribute requirements and tests; and any supporting RDL sections referenced by the requirements descriptions (e.g., definition of the 'user-commands' of Figure 6(a) and 'scg-ftl-ac' of Figure 6(b). At the end of the requirements specification, documents referenced by the requirements specification body that are not expressed in RDL are found.

Requirements Definition Language Characteristics

The Requirements Definition Language is based upon the same syntactical constructs as the Problem Statement Language (PSL) of the ISDOS project.[9] The language consists of *objects, relationships, descriptors,* and *associators.* Objects are essentially equivalent to nouns in English—they represent the things being described (CIPSSM primitives and associated information elements) when describing the CIPS. An *object-type* is a generic class of objects. The object-types are categorized according to what aspect of the RSM and CIPSSM they support. In Figure 6(a) the object 'scg-functions' is an example of the object-type FUNCTION.

Relationships are the "verbs" of RDL—they define the as-

(a)

```
DEFINE FUNCTION                    scg-functions;
    USES              user-commands;
    DERIVES           scg-output;
    UPDATES           data-base;
    PERFORMED BY      modified-scg-system;
    UTILIZED BY       scg-operation;
```

(b)

```
DEFINE FUNCTION-TEST               scg-function-testl;
    TESTS             scg-functions, user-activities;
    DESCRIPTION;
       This test sequence is designed to provide
       a customer acceptance test for creating
       structure charts, as one of the necessary
       functions of the modified-scg-system.;
    INITIAL-CONDITIONS ARE         scg-ftl-1, user-cmd-seq;
    FINAL-CONDITIONS ARE           scg-ftl-2;
    ACCEPTANCE-CRITERIA ARE        scg-ftl-ac;
    PROCEDURE;
       DO.
         set initial-conditions to 'scg-ftl-1'.
         DO UNTIL final-conditions='scg-ftl-2'.
            execute initial-condition
                    'user-cmd-seq'.
         ENDDO.
         check acceptance-criteria 'scg-ftl-ac'.
       ENDDO.;
```

Figure 6—(a) A function requirement extracted from the initial CIPS description shown in Figure 3
(b) A test for the function requirement

sociations between objects based upon the allowed relationships between object-types as determined by the CIPSSM and the RSM. In Figure 6(a) USES is an example of a relationship associating 'scg-functions' and 'user-commands.' Relationships are designed to be complementary in the sense that if the object-types are interchanged in an RDL statement, an equivalent relationship can be formed.

Descriptors are RDL constructs which fall outside of the object, relationship character of most RDL statements. Descriptors are associated with objects but are not objects themselves. Descriptors are an important source of redundancy which is counted on to help reduce the gap between intent and designed behavior. The most common *descriptor-type* is the comment-entry. This descriptor-type consists of English language text, or any more formal language of the users' choosing, that can be used to further describe an object outside the realm of the object's relationships with other objects. Descriptors are associated with objects via *associators*. Associators and descriptors allow one type of extensibility within RDL by permitting any desired language (e.g., SARA's GMB expressions, program description languages) to be included in the RDL specification. The associator DESCRIPTION and its comment-entry is illustrated in Figure 6(b).

The RDL syntactical constructs are patterned after the constructs that are supported by the ISDOS Project's Metagenerator and Generalized Analyzer.[20] RDL *statements* are grouped together by *sections*. Each section defines an object name and the relationships of that object to all other objects in a specification, plus the descriptors associated with that object. Figure 3 illustrates simple RDL sections that describe CIPS views.

RDL implementation of CIPSSM

RDL implements the CIPSSM by providing constructs that allow all CIPSSM primitives and properties to be described, as well as maintaining a basis for the semantic rules. There are unique RDL object-types for primitives that exist in the environment domain versus the CIPS domain. Their purpose is to provide a mechanism for maintaining the universe partition in a CIPSSM description.

RDL support of RSM

RDL supports the RSM by providing constructs that allow the following: (1) the four CIPS views to be incrementally developed and then rigorously coupled (Figure 3), (2) easy extraction of requirements by category (Figure 6(a)), (3) association of tests with each requirement (Figure 6(b)), and (4) easy extraction of RDL sections for the organization of the requirements specification.

The RDL syntax permits the designer/customer to describe the objects and relationships associated with any one particular CIPS view and then later to add the relationships that couple the views.

REQUIREMENTS ANALYSIS TECHNIQUES TOOLS AND PROCEDURES

Analysis techniques have been defined as a set of checks on the information within the specification to determine its compliance to the criteria of being understandable, consistent, complete, traceable, testable, realizable, and allowing design freedom. These checks are driven by the CIPSSM rules, heuristics of the design activity, and the modelling and analysis power of SARA. The requirements definition activity is designed to be extensively supported by computer-aided tools. The tools can be categorized into five functional areas: (1) an RDL interpreter and editor (plus associated database storage mechanism), (2) database query, (3) analysis checkers, (4) graphics support of the CIPS view constructions and translation to corresponding RDL, and (5) CIPS view to SARA model construction and translation.

An eighteen-step procedure has been developed that serves as a guideline on how to construct a model of the CIPS using RDL and graphics, perform analysis checks, and decompose a specification.

USAGE EXPERIENCE

The ISDOS Project's META System[20] was used to implement an RDL interpreter and editor (including database management system) and query system. A collection of SARA tools (fully operational and accessible on the ARPANET) exist and include the GMB Simulator and Control Flow Analyzer, plus structural modelling tools and a sophisticated help system. Using this continually expanding support environment, RDL has been applied in two practical CIPS specification activities since early 1981. The experience of these applications revealed that the modelling scheme was well liked, particularly

for the separation of environment and CIPS, and the provision of integrated multiview aspects. However, the following improvements are considered essential:

1. A graphics interface is needed to reduce the tedium of CIPS view construction and translation into RDL.
2. A better user interface to the support environment is needed.
3. Automated utilities to support test construction are needed to lessen the difficulty of creating appropriate requirements tests.
4. The ease with which RDL can express specifications of existing building blocks (e.g., manufacturers chip specifications) must be tested.

REFERENCES

1. Ross, D. T., and K. E. Schoman, Jr. "Structured Analysis for Requirements Definition." *IEEE Transactions on Software Engineering,* Vol. SE-3, No. 1, January 1977.
2. Alford, M. W. "Requirements for Distributed Data Processing." *Proceedings of First International Conference on Distributed Data Processing,* IEEE, 1979.
3. Alford, M. W. "Software Requirements Engineering Methodology (SREM) at the Age of Two." *COMPSAC 78 Proceedings,* November, 1978.
4. Willis, R. R. "DAS: An Automated System to Support Design Analysis." *Proceedings of the Third International Conference on Software Engineering,* Atlanta, Georgia, May 1978.
5. Alford, M. W., and I. F. Burns. "R-nets: A Graph Model for Real-time Software Requirements." In *Proc. Symp. On Comput. Software Eng., MRI Symp. Ser.,* Vol. XXIV, Polytechnic Press, Brooklyn, NY.
6. Riddle, W. E., J. C. Wileden, J. H. Sayler, A. R. Segal, and A. M. Stavely. "Behavior Modeling During Software Design" *IEEE Transactions on Software Engineering,* Vol. SE-4, No. 4, July 1978, pp. 283–292.
7. Zave, P., and R. T. Yeh. "Executable Requirements For Embedded Systems." *Proceedings of the Fifth International Conference on Software Engineering,* San Diego, California, 1981.
8. Heninger, K. "Specifying Software Requirements For Complex Systems: New Techniques and Their Application." *IEEE Transactions on Software Engineering,* Vol SE-6, No. 1, January 1980.
9. Teichroew, D., and E. A. Hershy III. "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems." *IEEE Transactions on Software Engineering,* Vol. SE-3, No. 1, January 1977, pp. 41–48.
10. Gardner, R. I. "A Methodology for Digital System Design Based on Structural and Functional Modeling." Ph.D. dissertation in Computer Science, University of California, Los Angeles, January 1975.
11. Estrin, G. "Modeling for Synthesis—The Gap Between Intent and Behavior." *Proceedings of the Symposium on Design Automation and Microprocessors,* Palo Alto, California, February 24–25, 1977, IEEE, Piscataway, New Jersey, 1977, pp. 54–59.
12. Estrin, G. "A Methodology for Design of Digital Systems—Supported by SARA at the Age of One." *Proceedings of the National Computer Conference,* Anaheim, California, June 1978.
13. Winchester, J. W. "Requirements Definition and Its Interface to the SARA Design Methodology for Computer-Based Systems." UCLA Technical Report, UCLA-ENG-8092, January, 1981.
14. Razouk, R., M. Vernon and G. Estrin. "Evaluation Methods in SARA—The Graph Model Simulator." *1979 Conference on Simulation, Measurement and Modeling of Computer Systems,* Boulder, Colorado, August 1979.
15. Razouk, R. R. "Computer-Aided Design and Evaluation of Digital Computer Systems." UCLA Technical Report, UCLA-ENG-8055, February 1981.
16. Penedo, M. H. "The Use of a Module Interface Description in the Synthesis of Reliable Software Systems." UCLA Technical Report, UCLA-ENG-8091, January 1981.
17. Vernon, M., D. Patel, and G. Estrin. "A SARA Building Block Model: Am2909 Microprogram Sequencer." UCLA Internal Memorandum #210, October 1981.
18. Campos, I. M., and G. Estrin. "SARA Aided Design of Software for Concurrent Systems." *Proceedings of the National Computer Conference,* Anaheim, California, June 1978.
19. Razouk, R., and G. Estrin. "Modeling and Verification of Communication Protocols in SARA: The X.21 Interface." *IEEE Transactions on Computers,* Vol. C-29, No. 12, December 1980, pp. 1038–1052.
20. Teichroew, D. "Overview of the META System." ISDOS Research Project, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan, ISDOS Project META-1 Memorandum, May 1977.