View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

SYSTEM REQUIREMENTS LANGUAGE FOUNDATION FOR SOFTWARE ENGINEERING

Charles R. Everhart
Teledyne Brown Engineering
Huntsville, Alabama

ABSTRACT

"System Requirements Language" here refers to those languages (defined by a formal set of syntax rules and semantics) whose purpose is the explicit and comprehensive expression of system definition and design facts. Not only is a formal requirements language necessary in specifying precisely the functional and performance characteristics of the system at all levels of definition and design, but at the same time this information can be used to predict the costs in time and money required to develop, implement, operate, and maintain a proposed system. Other benefits derived from this information include the direct generation of system and environment models used in the analysis of design solutions, direct generation of test criteria to be used during the test and integration phases of system development and the providing of a vehicle for maintaining configuration control throughout the life cycle of the system. "System" refers to not only software systems developed with "Software Development Methodologies", it also refers to the methodologies themselves.

1. MOTIVATION

According to a number of articles written recently [4,6] the costs of software development are becoming dominant (i.e., 50 to 90 percent of the cost of future data processing systems). In spite of the proliferation of programming languages [1] and software development techniques devised during the past 15 to 20 years, progress in reducing the costs of software has been disappointing. One source has indicated a decrease in productivity over this same period [3].

After a cursory analysis of major software developments, it appears that the industry has been attacking the wrong problem. A very large percentage of software development costs are associated with the definition, design, testing, and integration activities. The development of programming languages and techniques has been directed mainly toward the coding activity which represents only a small percentage of software development (as low as 17% for a large 7.5 year project [6]).

If the software industry is expecting to see large reductions in the cost of software, it must attack those problems connected with definition, design, testing and integration activities representing anywhere from 45 to 85 percent of most data processing development costs.

The first step in attacking the cost of these activities appears to be the development of a precise, yet convenient, system requirements specification and analysis language. Teledyne Brown Engineering has been actively involved in this type of development since 1971 and has devised a requirements language called IORL (Input/Output Requirements Language) which claims the objectives of the preceding discussion.

2. IORL

IORL is a formally defined language (syntactically and semantically [2,5] which uses a combination of both graphic symbols and mathematical notation to express system definition and design ideas. Block diagrams (analogous to those used in control theory) organized in a hierarchical manner identify the parts of a system and the interfaces between these parts at all levels of system definition and design.

Descriptions of each interface identified are contained in a set of tables called IOPT's (Input/Output Parameter Tables). Another diagram called in "IORTD" (Input/Output Relationships and Timing Diagram) is used to define the total transformation function from input to output as well as the response time requirements for each and every block in the hierarchy. These diagrams (analogous to a "Transfer Function" in control theory) provide the symbols for specifying the sequential, simultaneous, logical, mathematical and time requirements between inputs and outputs of each given block. The elements of IORL and hierarchical structure of requirements information are characterized in Figures 1 and 2.

3. IORL STORAGE AND RETRIEVAL FACILITY

Storage, retrieval and modification of IORL diagrams and tables has been implemented on a stand-alone PDP/11 based graphics terminal (GT44 and GT46) with 16K memory. The interactive graphics system includes a 17-inch refresh type graphic screen with lightpen capability as well as an electrostatic printer plotter which produces 8.5 x 11 inch copies of the screen. In edit mode, IORL information is entered by pointing the lightpen at a location on the screen and then pressing the keyboard button associated with the desired symbol. In display mode, system details are accessed by directing the lightpen to points of interest on the higher level diagrams of the hierarchy. This results in the subsequent display of these details on the screen. Requirements diagrams are stored on and retrieved from disk packs which are also a part of the facility.

4. BALLISTIC MISSILE DEFENSE (BMD) PARTITIONING STUDY EXAMPLE

Not only is a formal requirements language necessary in specifying precisely the functional and performance characteristics of the system at all levels of definition and design, but at the same time this information can be used to predict the costs in time and money required to develop, implement, operate and maintain a proposed system. Other benefits derived from this information include the direct generation of system and environment models used in the analysis of design solutions, direct generation of test criteria to be used during the test and integration phases of system development and the providing of a vehicle for maintaining configuration control throughout the life cycle of the system.

In an attempt to determine the feasibility of the preceding thesis, a study entitled "BMD Partitioning Study", was performed. The objective of this study was to demonstrate that certain quantitatiave characteristics, related to system development and operation costs, could be derived directly and mechanically from formally defined system requirements specifications. IORL was used as the language for specifying the definition and design requirements.

The demonstration consisted of four basic steps. First, the BMD system requirements and its environment were defined using the complete set of IORL symbols, tables and diagrams. This information, which represented the first level in the system hierarchy, was the only infor-

mation used in the subsequent requirements analysis and design activities. After checking the first level specification for completeness and consistency, the second step involved designing two different solutions to the BMD requirements, again expressing these solutions in IORL and placing this information in the second level of the system hierarchy. The purpose in specifying two design solutions was to compare the total system costs which resulted from each of the solutions. In the third step, each completely specified solution was validated against the BMD requirements by exercising the environment, BMD, and solution models (all written in IORL) and then comparing responses at the BMD/environment interfaces.

In this manner it was established that each solution responded to the environment exactly as required by the BMD requirements (model). If not, the design solution was corrected. In the final step, each solution was evaluated to determine its effect on total system costs. This evaluation, which was a combination of static and dynamic analysis of only that information contained in the IORL specifications, produced the summary partition evaluation results shown in Figure 6. These summary results were derived from a series of intermediate results which plotted for example bandwidth for each interface as a function of time, storage required as a function of time, functional speed required, etc.

Our experience with this study has resulted in the following conclusions:

- Quantitative measures related to the costs of a system can be determined from an analysis of system definition and design requirements.
- The requirements language used to specify system definition and design facts is the key factor in the success of the preceding demonstration (i.e., the language must have certain characteristics and enforce certain disciplines).
- The proper specification of definition and design requirements can provide information necessary to all phases of a system development (definition, design, implementation, test and integration).

5. FUTURE R&D ACTIVITIES

Teledyne Brown Engineering has plans to continue the development of IORL related techniques and tools. The immediate future calls for the development of the following computer utility packages:

- An extensive IORL diagnostic package (syntax analyzer)
- A set of configuration management tools
- An expended set of graphic editing features for the storage and retrieval system
- A utility program library
- A set of functional and analytic simulation compilers which will transform IORL information into FORTRAN or PASCAL simulation statements.

We have also experimented with the direct generation of assembler source code (Marco-II Assembler) from IORL information and have determined that the information content of IORL will support the development of a set of IORL compilers. The major problems associated with this last activity are the implementation of mathematical functions so easily represented in IORL.

REFERENCES

1. Shea, William E., "DOD-1, A Common Language", ISRAD in Touch, Volume 2, No. 1, February 1978.

2. Everhart, C. R., "IORL Analysts Users' Manual", Section 1 – Syntax, Teledyne Brown Engineering, May 1977.

3. Dolotta, T. A., et al, "Data Processing in 1980-1985", John Wiley and Sons, 1976.

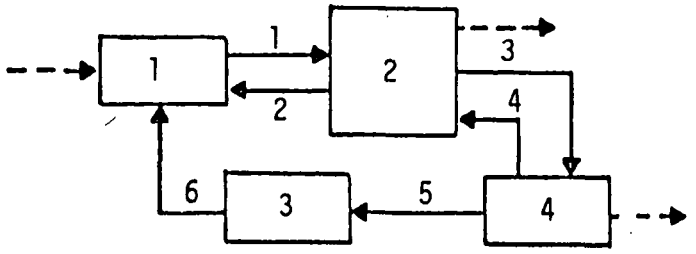
4. Boehm, B. W., "Software and Its Impact: A quantitative Assessment", Datamation, pp. 48-59, May 1973.

5. Everhart, C. R., "IORL Analysts Users' Manual", Section 2 – Semantics, Teledyne Brown Engineering, May 1977.

6. Ramamoorthy, C. V., et al., "Software Requirements and Specifications: Status and Perspectives", Engineering Research Recommendations, Draft Appendix A, August 12, 1977.

IORL ELEMENTS

1



SBD

2

IOPT 1

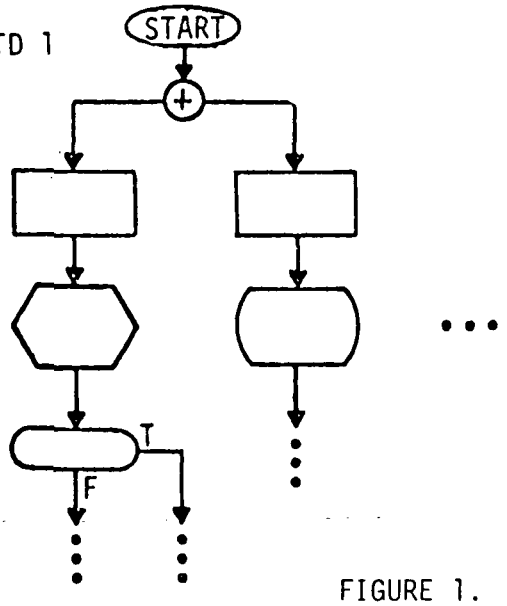
G	PARAMETER	VALUES	UNITS

IOPT 6

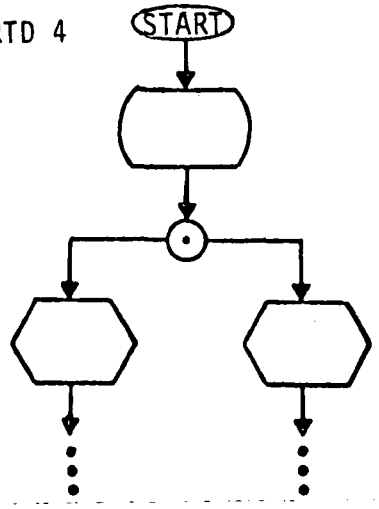
IOPT's

3

IORTD 1



IORTD 4



IORTD's

FIGURE 1. IORL ELEMENTS

C. Everhart
Teledyne Brown Eng.

IORL INFORMATION HIERARCHY DIAGRAM

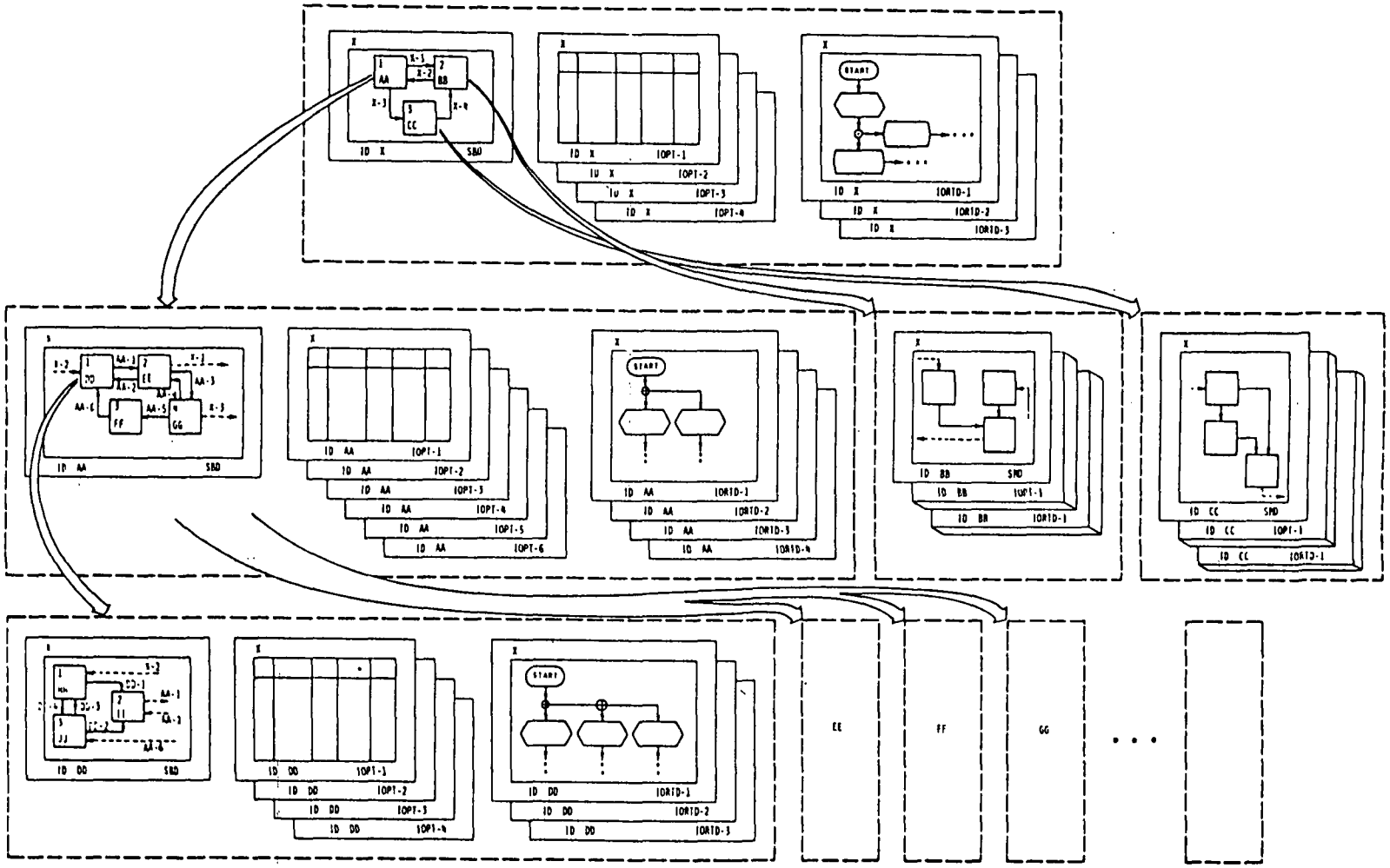


FIGURE 2. IORL INFORMATION HIERARCHY DIAGRAM

C. Everhart
Teledyne Brown Eng.

CURRENT CAPABILITIES – STORAGE AND RETRIEVAL
OF REQUIREMENTS USING INTERACTIVE GRAPHICS

- BASIC STORAGE AND RETRIEVAL OF INFORMATION
  - ▲ DISK PACKS (1200 PAGES/PACK)
  - ▲ CLASSIFIED FACILITY
- ON LINE EDITING
  - ▲ CRT
  - ▲ LIGHT PEN
  - ▲ FUNCTION KEYS
- HARDCOPY
  - ▲ REPORT QUALITY
- DOCUMENTATION AUDIT
- MERGE SYSTEM PAGES
- AUTOMATIC ACCOUNTING

Figure 3. Current Capabilities – Storage and Retrieval of Requirements Using Interactive Graphics

C. Everhart
Teledyne Brown Eng.

SYSTEM CONFIGURATION

- PDP-11/40 WITH 16K MEMORY
- 17" GRAPHIC DISPLAY WITH LIGHTPEN
- TELETYPE
- 2 DISK DRIVES
- ELECTROSTATIC PRINTER/PLOTTER (8½ X 11 FAN FOLD)

Figure 4. System Configuration

C. Everhart
Teledyne Brown Eng.

COMPUTERIZED ANALYSIS (PARTITIONING STUDY)

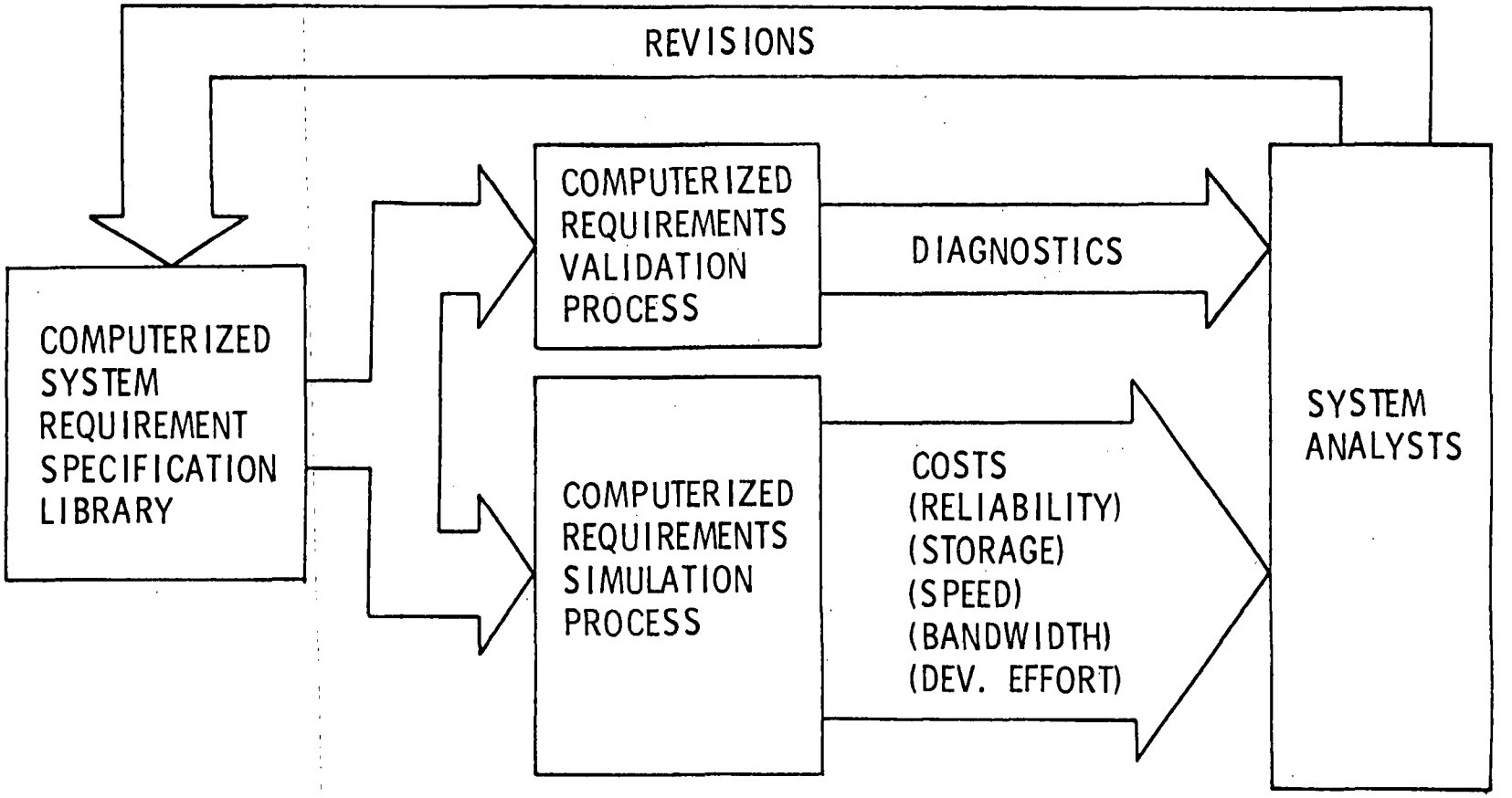


FIGURE 5. COMPUTERIZED ANALYSIS (PARTITIONING SYSTEM)

PARITITION EVALUATION RESULTS

	PARTITION 1				PARTITION 2			
	GND	TRKER	SENSOR	TOTAL OR WORST	GNDSYS	STTCOR	SENS	TOTAL OR WORST
RELIABILITY	0.975	0.900	0.975	0.941	0.980	0.980	0.980	0.941
STORAGE (KBITS)	23.5	389.3 (2)	2.67	804.8	138.6	284.4	27.9	432.7
PROCESS SPEED (η SEC)	5.44	5.85	5.56	5.44	0.814	0.814	0.814	0.814
PROGRAMMING COMPLEXITY (MAN-MONTHS)				35.65				26.3
PEAK BANDWIDTH (MBITS/SEC)				2.52				20.94

FIGURE 6. PARTITION EVALUATION RESULTS

C. Everhart
Teledyne Brown Eng.
Page 11 of 13

C. Everhart
Teledyne Brown Eng.

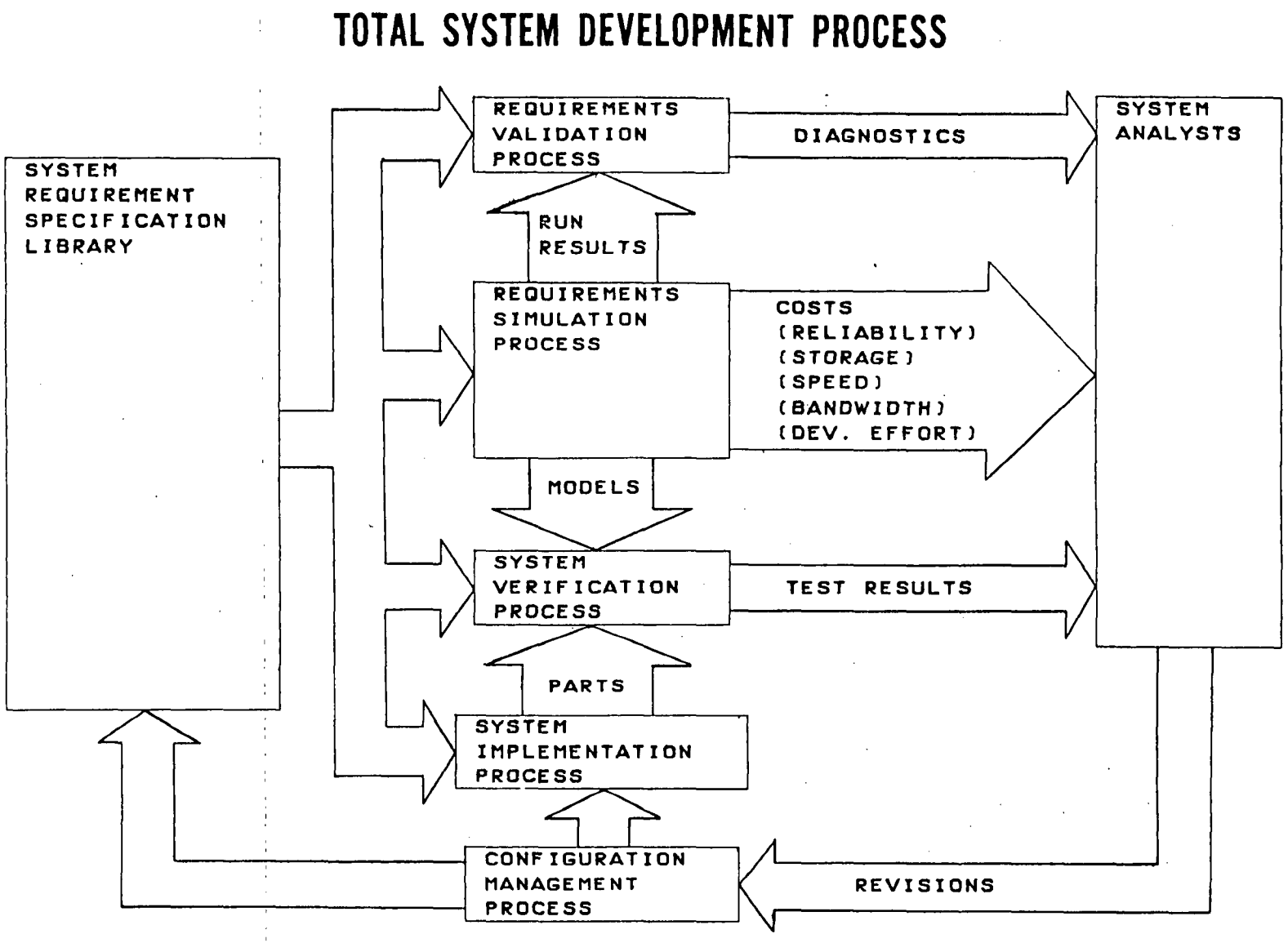


FIGURE 7. TOTAL SYSTEM DEVELOPMENT PROCESS

FUTURE IORL RESEARCH AND DEVELOPMENT

- DIAGNOSTICS PACKAGE (SYNTAX ANALYZER)
- CONFIGURATION MANAGEMENT TOOLS
- EXPANDED EDITING FEATURES (GRAPHICS)
- UTILITY PROGRAM LIBRARY (STATIC ANALYSIS)
- SIMULATION COMPILER
  - ▲ FUNCTIONAL: "IORL" TO "MODELER" TRANSLATOR
  - ▲ ANALYTIC: "IORL" TO "FORTRAN" TRANSLATOR
- COMPILER
  - ▲ "IORL" TO "FORTRAN" TRANSLATOR
  - ▲ "IORL" TO "PDP-11" MACRO-ASSEMBLER" TRANSLATOR

Figure 8. Future IORL Research and Development

C. Everhart
Teledyne Brown Eng.