Arts—An Automated Requirements Traceability System

Merlin Dorfman* and Richard F. Flynn**

Lockheed Missiles & Space Company, Inc.

Management of the hundreds or thousands of requirements associated with a large aerospace system is a tedious, error-prone process. Automated Requirements Traceability System (ARTS) has been developed to automate this process. ARTS is a bookkeeping program that operates on a data base consisting of system requirements and their attributes. It provides upward and downward traceability in a hierarchical structure, as well as data base management and output operations on any requirement-related attributes selected by the user. ARTS is currently implemented on the UNIVAC 1110, DEC VAX-11/780, and IBM computers.

I. INTRODUCTION

Over the past decade and more, phrases such as "software crisis" [1] and "software tar pit" [2] have been used to describe a syndrome of now familiar problems in the development of large data systems. The syndrome includes excessive costs, schedule overruns, unsatisfactory performance, and difficulty of maintenance. [3–6]. Intense effort has been directed to finding the causes of these problems and to making software development into a science capable of consistently generating a high-quality product. [3,7–9].

These investigations have determined that requirements deficiencies are among the most important contributors to the problems:

> In nearly every software project which fails to meet performance and cost goals, requirements inadequacies play a major and expensive role in project failure [10].

> There are four kinds of problems that arise when one fails to do adequate requirements analysis: (a) Top-down design is impossible, (b) testing is impossible, (c) the user is frozen out, and (d) management is not in control [11].

The investigations also show that the cost of correcting requirements-related problems increases drastically with time:

> Clearly, it pays off to invest effort in finding requirements errors early and correcting them in, say, 1 manhour rather than waiting to find the error during operations and having to spend 100 manhours correcting it. [12].

Requirements engineering has emerged as a subspeciality in systems development. Standards and practices for this subspecialty include methods for determining top-level requirements through discussions with the prospective customer, establishment of a top-down hierarchy for the system [14], allocation and flowdown of requirements to lower levels [14,15], and maintenance of requirements traceability upward and downward at all levels in the hierarchy. [3,16,17].

Although the need for traceability of requirements is not disputed, few projects establish traceability, and even fewer do so properly. One important reason is the format of the conventional specification. Each prose paragraph may contain several requirements; each specification element contains many such paragraphs [18]. All the requirements for a given system element are traceable upward to one element at the next higher level and downward to several elements at the next lower level. But traceability is not established between individual requirements. Another obstacle is the sheer volume of requirements which can number into the thousands for large systems. Computer automation of traceability was conceived as an approach to make practical the establishment and maintenance of traceability for requirements hierarchies of any size [17–19].

Lockheed Missiles & Space Company, Inc. (LMSC) has produced an Automated Requirements Traceability System (ARTS). ARTS has been in limited use since June 1980 and in full operation since Oc-

---

*Senior Staff Engineer, Data Systems Technology.
**Staff Engineer, Data Systems Engineering.

Address correspondence to Dr. Merlin Dorfman, Lockheed Missle and Space Co., Inc., 1111 Lockheed Way, Dept. 62-HO, Bldg. 541, Sunnyvale, CA 94086.

The Journal of Systems and Software 4,63–74(1984)

0164-1212/84/$3.00

tober 1980. Automation, as expected, has made traceability a practical reality. Several other benefits have resulted from the use of ARTS:

1. ARTS permits the user to include any other requirements-related attributes of his choice in the computer data base. Normal data base management operations can be performed on these attributes. The ability to store and track attributes has provided a large number of anticipated and unanticipated benefits.
2. A commitment to traceability imposes a discipline on the requirements analyst that results in a better set of requirements even before a data base is generated.
3. ARTS permits a speed of response to changes in requirements (both in terms of manhours and elapsed time) that is not possible using manual methods.
4. The availability of rapid, accurate traceability information makes it easier for the engineer to detect inconsistencies and omissions in the requirements allocation and flowdown.
5. Although motivated primarily by software needs, ARTS is fully applicable to hardware or composite systems as well.

Like any good tool, ARTS not only makes it possible to do the existing job more efficiently, it makes improvements possible that could not exist without the tool. ARTS changes the systems engineering process.

II. GENERAL DESCRIPTION

A. Objectives and Configuration

ARTS[1] is a bookkeeping program that operates on a data base consisting of the *requirements* for a target system and *attributes* related to each requirement.

The major function of ARTS is to provide rapid and accurate *traceability,* upward and downward, in a requirements *hierarchy* or *tree.* The purposes of maintaining traceability are:

1. To verify that requirements *allocation* and *flowdown* have been done correctly, i.e., that every *top-level* requirement has been allocated to one or more *system elements* through generation of lower level requirements, and that every *lower level* requirement is properly derived from the top level.
2. To permit assessment of the impact of changes. If a top-level requirement is changed by the customer, all affected lower level requirements can be isolated. If a lower level requirement must be changed (e.g., because of some design consideration), the higher level requirements from which it is *derived* can be isolated so the effects on performance can be assessed.
3. To provide the basis for generation of complete test plans and procedures. To test against requirements, the requirements allocated to each system element are addressed in tests for that element.

The existence of the ARTS software and requirements data base provides other advantages at little or no additional cost. The ability to store attributes (e.g., source documentation reference, schedule dates, or responsible persons) permits rapid access, manipulation, and output of the attributes, as compared to manual methods. The data base aids *configuration management,* since it is a single, authoritative source of current status. Change history can be maintained by recording copies of all updated records.

ARTS is hosted on the UNIVAC 1110, DEC VAX-11/780, and various IBM computers, all with identical functional capabilities. The software currently consists of approximately 6200 lines of executable FORTRAN IV code. Of this, about 3500 lines are from the Regional Information Management System (RIMS) data base manager. RIMS is a relational data base manager written by Lockheed Engineering and Management Services Company for NASA-Johnson Space Center; with NASA permission LMSC uses RIMS in ARTS and in other applications. Minor modifications were made to about 300 lines of RIMS to tailor it for the ARTS application, and an additional 2700 lines of ARTS-specific code were generated. About 95% of the ARTS code is common between the UNIVAC (EXEC VIII), VAX (VMS), and IBM (VS, MVS, and VM) versions. The differences are in machine-dependent functions such as string processing, file I/O, and word size. Memory requirement is about 50,000 words on the UNIVAC with no overlays required.

B. Data Flow and Input

Figure 1 shows the overall ARTS data flow.

Inputs consist of requirements data (text and attributes); formats for data storage, for batch input and update, and for outputs; and commands to both the computer operating system and the ARTS software. ARTS stores the requirements data according to the specified formats, retrieves data from the files, performs manipulations on it as specified by the commands, and formats it for output. Output consists of a variety of reports routed to the computer terminal, line printer, or special devices. Output examples are given in Section IID.

[1]Section V is a glossary of definitions of terms. The first occurrence of each term in the text of this section is italicized.

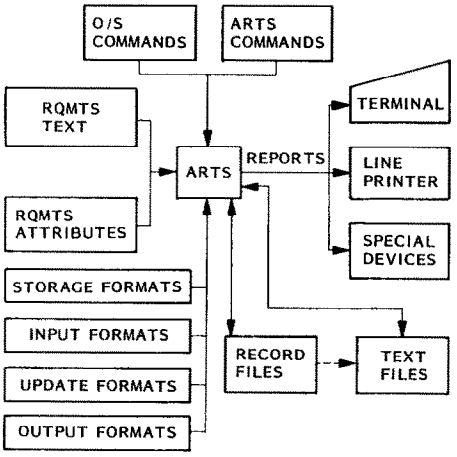


Figure 1. Overall ARTS data flow.

Figure 2 elaborates on the input processing. Either realtime inputs or batch data can be entered at the computer terminal and edited using the system editor. Batch data also can be input in three other ways: from cards, from tape, or by direct transfer from a word processor using special hardware. Batch data normally are entered into an edit file so that corrections can be made if needed. ARTS is then run against the corrected file. The edit file also can be converted from VAX to UNIVAC format or vice versa if necessary.

Control of processing is exercised by means of commands. Commands, like data, may be entered at a terminal or via a batch runstream; they consist of two-character mnemonics specifying the desired operation, followed by any other required control information.

C. Requirements Data Base

This section describes the relationship between requirements information and the ARTS data base. The relationship is illustrated using an example system and its requirements.

Figure 3 shows the hierarchy of the example system, with level 1 being the system, level 2 being the segment, 3 the subsystem, 4 the component, and 5 the subcomponent. Shown with each element of the hierarchy are the requirements to be met by that element. The traceability is element-to-element, as would be found in a conventional specification, e.g., the requirements allocated to subsystem E are known to be derived from those of segment C.

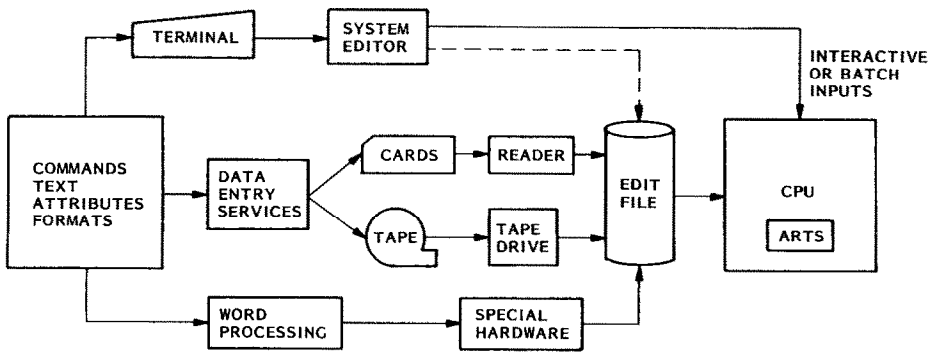
Figure 4 shows requirements trees including individual traceability. The lines indicate the derivation (upward) and breakdown (downward) relationship between requirements. Note that there are two instances (SSX001 and SSX002) of multiple derivations for one requirement, which occasionally occurs.

In both Figures 3 and 4, requirements are represented by arbitrary identifiers called requirements identification or REQID. REQID is the unique primary key; there must be exactly one per requirement. REQID must be an attribute included in the data base design and entered for every defined requirement. It has proven beneficial to construct REQID in a mnemonic form (e.g., beginning wtih letters representing the system element to which the requirement is assigned and ending with a sequence counter of assignments to that element). This practice has been followed in the example hierarchy. For example SSF003 is the (REQID of the) third requirement assigned to subsystem F.

Table 1 is a partial list of information such as would be entered into an ARTS data base to represent the system described in Figures 3 and 4. Besides REQID, level and derivation are shown explicitly, and other attributes and text are implied.

The derivation attribute defines the structure of the requirements hierarchy; ARTS uses it to provide traceability information. The user therefore is expected to select derivation as an attribute and to include at least a temporary entry for each defined requirement. If derivation is blank for a particular requirement, ARTS assumes that requirement to be top level.

Figure 2. Input processing flow.



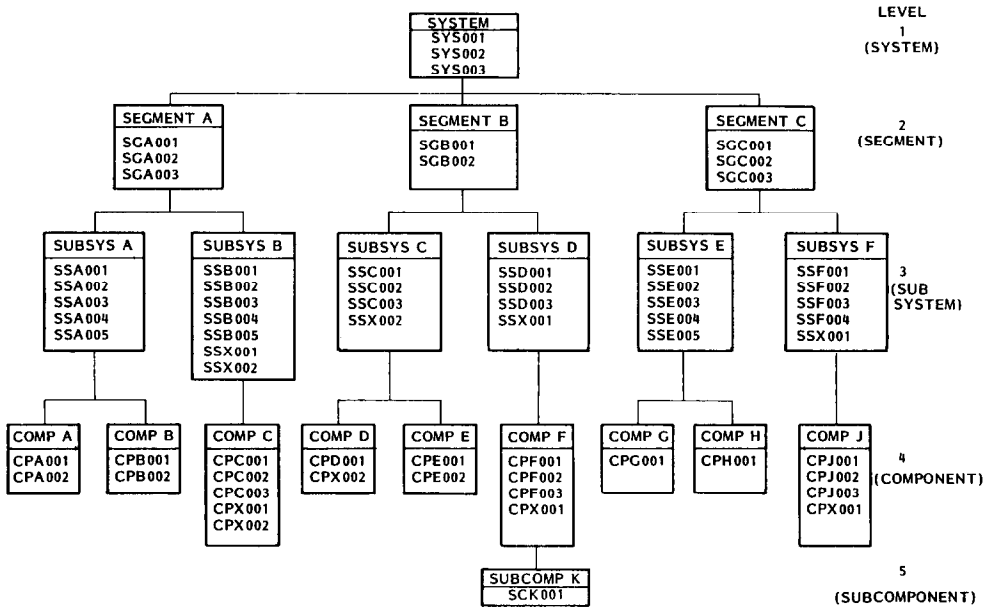


Figure 3. Example system hierarchy showing requirements flowdown.

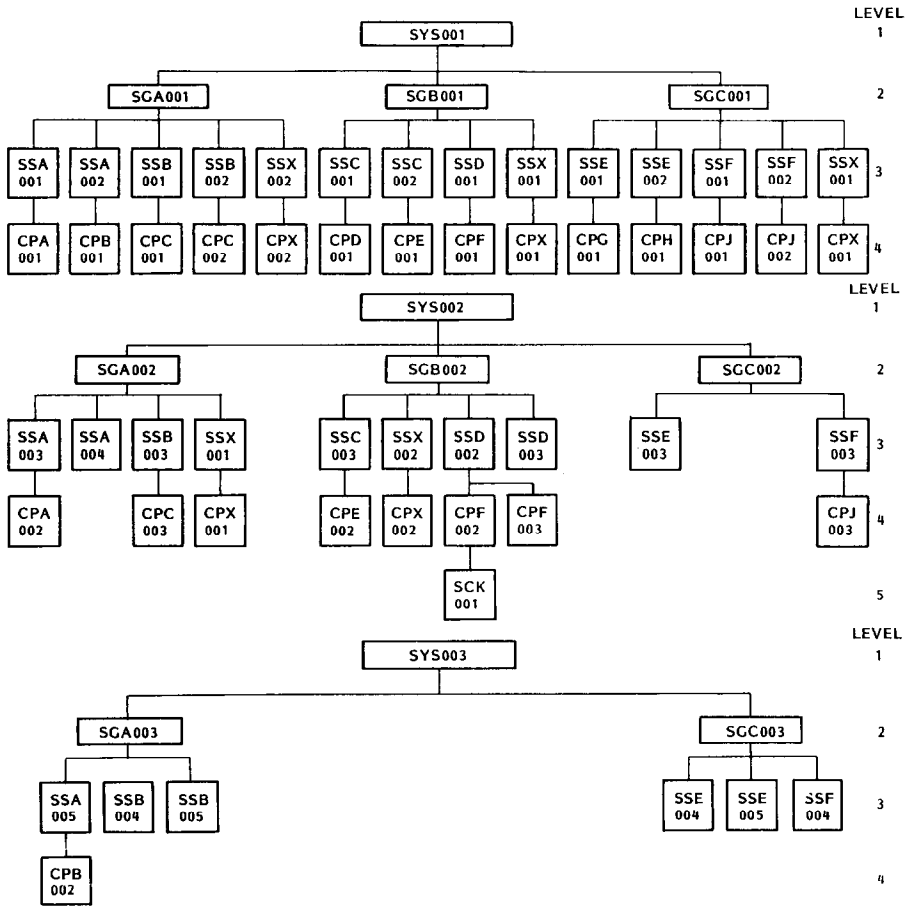


Figure 4. Requirements trees for example hierarchy.

Table 1. Partial ARTS Data Base Contents Corresponding to Hierarchy of Figures 3 and 4

REQID	Level	Derivation	Other Attributes	Text
SYS001	1	—	xxxx	xxxx
SYS002	1	—	xxxx	xxxx
SYS003	1	—	xxxx	xxxx
SGA001	2	SYS001	xxxx	xxxx
SGA002	2	SYS001	xxxx	xxxx
SGA003	2	SYS001	xxxx	xxxx
SSA001	3	SGA001	xxxx	xxxx
SSA002	3	SGA001	xxxx	xxxx
SSB001	3	SGA001	xxxx	xxxx
CPA001	4	SSA001	xxxx	xxxx

Level is implicit in the REQID-derivation data base and need not be inlcuded in the format or actually entered. It is used only to provide level numbers in various outputs (and for manipulations as requested by the user). Consistency between the entered level and that implied by REQID and derivation is not checked.

The user may include in his data base design any other attributes he chooses; however, information need not be entered in any field if, for example, it is not currently available. If needs change, the data base can be restructured to exclude existing attributes, to include new ones, or to change names or field lengths.

Various ARTS users have included in their data bases such attributes as responsible engineer, date of origination, revision number, work breakdown statement paragraph, source document (e.g., spec) paragraph leading to the requirement, report paragraph that describes how the requirement is met, a key word that categorizes the requirement (so that, for example, all requirements relating to "power consumption" or "pointing accuracy" can be accessed), interfaces to other requirements or elements, test plan references, and schedule dates. Many benefits result from data base operations (select, sort, etc.) and reports based on these attributes.

D. Outputs

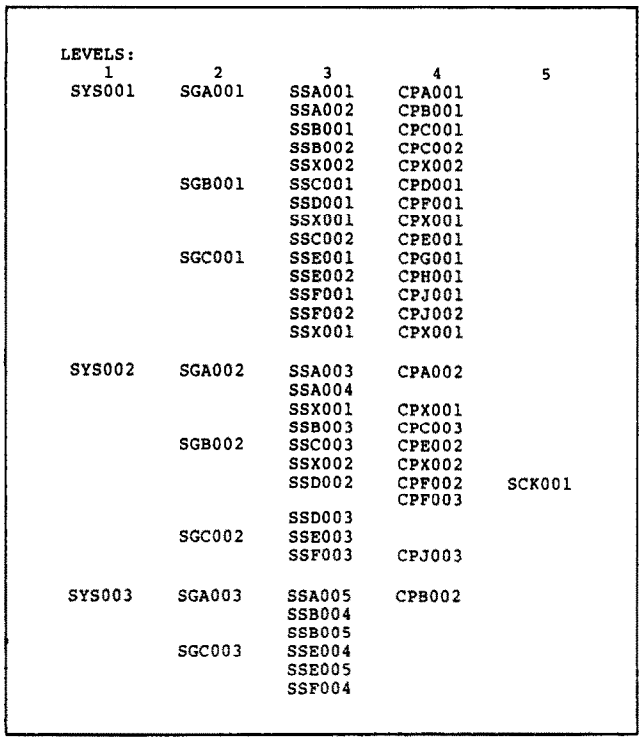
Figure 5 shows full ARTS output of the contents of three records from the above data base. They are included to show the actual data base design (format)

```
---------------------------------------------------------------------------
REQID               SYS002
LEVEL               01
SOURCE              3.2.2.1
DERIVATION
KEYWORD             FDA
STATUS              A
SUMMARY             ALL PROCS. MUST BE DEFINED AND SUBMIT BY 01/01/83
RESP ENGR           J. SMITH
PHONE               23456
REV                 3
DATE                010181
TEXT                B101.DAT
ALL PROCESSES WILL BE OUTLINED AND SUBMITTED TO THE "BETTER PROCESSING
TECHNOLOGY REVIEW BOARD" OF THE FOOD AND DRUG ADMINISTRATION BY JUNE 1,
1983 FOR APPROVAL.  AN AD HOC AUDIT TASK FORCE WILL LATER BE ESTABLISHED
TO VERIFY CONTINUED COMPLIANCE.
---------------------------------------------------------------------------

---------------------------------------------------------------------------
REQID               SGA002
LEVEL               02
SOURCE              3.2.2.2
DERIVATION          SYS002
KEYWORD             TRANSPORT
STATUS              P
SUMMARY             PRODUCT TRANSPORTATION SHALL BE DEFINED, REPORTED
RESP ENGR           J. JONES
PHONE               12345
REV                 1
DATE                020281
TEXT                C103.DAT
PRODUCT TRANSPORTATION, FROM FIELD LOCATION TO MANUFACTURING SITE
(INTER AND INTRA-PLANT MODES) SHALL BE IN A DEFINED MANNER WITH
COMPLETE ACCOUNTING CONTROLS.  ALL CRITICAL POINTS WILL BE HIGH-
LIGHTED AND RECORDED.
---------------------------------------------------------------------------

---------------------------------------------------------------------------
REQID               SSA003
LEVEL               03
SOURCE
DERIVATION          SGA002
KEYWORD             SANITATION
STATUS              A
SUMMARY             FUNGUS MUST BE CONTROLLED TO MET FDA MONORITING
RESP ENGR           J. BROWN
PHONE               54321
REV                 2
DATE                030381
TEXT                D114.DAT
CAN BELTS ARE TO BE SWABBED ON REGULAR INTERVALS FOR FUNGUS
CULTURES.  FDA USES FUNGUS AS AN INDEX TO SANITATION LEVELS.
---------------------------------------------------------------------------
```

Figure 5. Contents of data base records corresponding to REQIDs SYS002, SGA002, and SSA003.



```
LEVELS:
   1        2        3        4        5
 SYS001   SGA001   SSA001   CPA001
                   SSA002   CPB001
                   SSB001   CPC001
                   SSB002   CPC002
                   SSX002   CPX002
            SGB001 SSC001   CPD001
                   SSD001   CPF001
                   SSX001   CPX001
                   SSC002   CPE001
            SGC001 SSE001   CPG001
                   SSE002   CPH001
                   SSF001   CPJ001
                   SSF002   CPJ002
                   SSX001   CPX001

 SYS002   SGA002   SSA003   CPA002
                   SSA004
                   SSX001   CPX001
                   SSB003   CPC003
            SGB002 SSC003   CPE002
                   SSX002   CPX002
                   SSD002   CPF002   SCK001
                            CPF003
                   SSD003
            SGC002 SSE003
                   SSF003   CPJ003

 SYS003   SGA003   SSA005   CPB002
                   SSB004
                   SSB005
            SGC003 SSE004
                   SSE005
                   SSF004
```

Figure 6. Downward tree for data base corresponding to Figure 4.

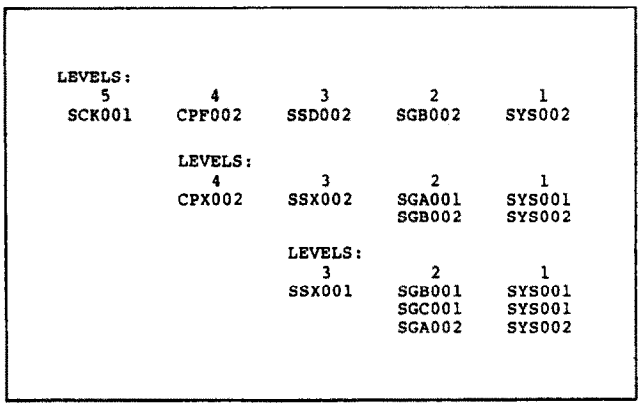
used, to illustrate some typical attributes and their values, and to demonstrate this particular report (output format) capability.

Figure 6 shows three downward tree reports corresponding to the example. The tree is arranged horizontally with each column representing a level. All the requirements in a column between two entries in the next column to the left are derived from the upper of those two (e.g., SSA001, SSA002, SSB001, SSB002, and SSX002 are derived from SGA001). Note that requirements with multiple derivations (and all deriving from them) are repeated, once for each derivation. If too many levels exist for the printer width, continuation marks are printed, and the lower levels are given later.

Figure 7 shows three representative upward trees. The upward tree runs from the input REQID (at the left) up to level 1. Note that multiple derivations are handled by branching to multiple rows at the correct point and continuing upward for each row.

The format of Figure 5 requires many lines per requirement. For a particular application, it may not be necessary to list everything in the record. If the information needed can be fitted into one line, a summary report can be obtained requiring only one line per requirement. Figure 8 is a sample of such a report. The format is constructed by the user.

Figure 9 is an example of a report in which the REQIDs of all requirements that meet particular criteria



```
LEVELS:
   5        4        3        2        1
 SCK001   CPF002   SSD002   SGB002   SYS002

            LEVELS:
              4        3        2        1
            CPX002   SSX002   SGA001   SYS001
                              SGB002   SYS002

                     LEVELS:
                       3        2        1
                     SSX001   SGB001   SYS001
                              SGC001   SYS001
                              SGA002   SYS002
```

Figure 7. Example of upward tree.

REQID	LEVEL	DERIVATION	SUMMARY
SYS001	01		REGS. MUST BE STRICTLY ADHERED TO, PASSIVE VIOLATIONS WILL NOT BE TOLER
SYS002	01		ALL PROCS. MUST BE DEFINED AND SUBMIT BY 01/01/83
SYS003	01		SHELF-LIFE SHALL BE DEMONSTRATED TO EXCEED 15 MONTHS @ 80 DEGREES F.
SGA001	02	SYS001	ALL FDA REGULATIONS SHALL BE INTERPRETED BY THE IN-HOUSE FDA INSPECTOR.
SGA002	02	SYS002	PRODUCT TRANSPORTATION SHALL BE DEFINED, REPORTED
SGA003	02	SYS003	STORAGE FACILITIES ARE TO BE OF REASONABLE CONTRUCTION, PROVIDING WEATH
SGB001	02	SYS001	ALL USDA REGULATIONS SHALL BE INTERPRETED BY THE IN-HOUSE USDA INSPECTOR
SGB002	02	SYS002	EACH MEMBER OF THE PRODUCT-SIZE MATRIX SHALL HAVE A WRITTEN RETORT PROCE
SGC001	02	SYS001	EPA STANDARDS SHALL BE MET ACCORDING TO PUBLICATION (CUSTOMER) 3-X
SGC002	02	SYS002	CONTAINERS SHALL BE CONSIDERED A CRITICAL POINT OF ASEPTIC MANUFACTURING
SGC003	02	SYS003	SHELF-LIFE STUDIES SHOULD CONCLUDE THAT PRESCRIBED MANUFACT. TECH. WORKS
SSA001	03	SGA001	SAFETY GUIDELINES ARE TO BE WRITTEN AND SUBMITTED BY AUGUST 14, 1982.
SSA002	03	SGA001	LABELS SHALL CONFORM TO FDA AND STATE LABELING LAWS.
SSA003	03	SGA002	FUNGUS MUST BE CONTROLLED TO MET FDA MONORITING

Figure 8. Example of one-line summary report.

are listed in a block format. In this case, all requirements having a level field equal to 2 were listed.

Figure 10 shows the parent–child report, designed to aid in evaluating requirements allocation. A selected requirement, SGB002, is input; its REQID, level, and text (or summary) are listed. Then the REQID, level, text (or summary), and lower level count[2] are given for every requirement derived from the input one. This helps the systems engineer determine the validity of the flowdown of the input requirement. A similar report exists where the "parent" of the input requirement (the one from which the input is derived) and all others derived from the parent are listed.

Most of these reports can be sent to the computer terminal for interactive use, as well as to the printer.

III. USE OF ARTS

A. ARTS and Systems Engineering

From the standpoint of systems engineering, "ARTS is a validated systems engineering tool that integrates people, procedures, hardware, and software. It provides a documented structure to the requirements of a large system [20].

The first phase of top-down systems engineering is an iterative process involving definition of a system hierarchy, allocation of requirements to elements of the hierarchy, and preliminary design. ARTS is useful in speeding the process of reviewing the preliminary allocation so that revisions to the hierarchy and/or allocations can be done quickly and in a small number of iterations. When source documentation references are included in the data base, ARTS helps tie the requirements to the spec or other customer-furnished requirements.

In each iteration, a large number of requirements will be generated and flowed down to lower levels. The total number of requirements increases rapidly with the number of levels, and efficiency in getting information into and out of ARTS is important.

A requirements definition form, such as shown in Figure 11, has been found useful for ARTS input. It agrees with the current configuration of the ARTS data base (Figure 5). When an engineer identifies a new requirement, he fills out the form. When the requirement achieves preliminary approval, the form is passed to keypunch or word processing. Groups of new requirements are entered into the ARTS data base at appropriate intervals using the batch input capability. ARTS outputs, such as Figure 10, are used to review requirements. Revisions also are input via a requirements form and the batch update capability.

Figure 12 is a schematic of the use of ARTS in systems engineering. Generation of requirements by the customer and by systems engineering is shown; the data enter ARTS via keypunch and the requirements librarian. Initial output from ARTS is individual reports; the Requirements Allocation Handbook (RAH) is a more definitive product when some stability has been reached [21]. It is a listing, down to the level of stability, of trees and of text and a few attributes for each requirement. Changes to ARTS and its documentation are produced by the development group.

The systems engineer's task of assessing the quality of allocation and flowdown is very difficult, made more

[2]A listing of the number of requirements at each level below "child."

Figure 9. Output of all requirements that have level = 2.

```
LEVEL:02
-------------------------------------------------------
SGA001    SGB001    SGC001    SGA002    SGB002    SGC002
SGA003    SGC003
```

```
**********************************************************************
            *PARENT REQUIREMENT SGB002       LEVEL 02*
**********************************************************************
THERE SHALL BE A WRITTEN PROCEDURE DEFINING ALL ASPECTS OF RE-
TORTING ALONG WITH A FDA APPROVAL FOR EACH PRODUCT FOR EACH
CONTAINER SIZE.  ANY PRODUCT LOT WHICH WAS SUBJECT TO A
DEVIATION FROM THESE PROCEDURES SHALL BE ISOLATED UNTIL A FDA
WRITTEN RELEASE IS GRANTED.

   CHILD REQID SSC003                    LEVEL 03
BLANCHING METHODS DESCRIBED BY FDA DOCUMENT P222 MUST BE USED TO
DRIVE ALL AIR FROM PRODUCT AND TO CREATE NECESSARY INITIAL TEMP-
ERATURE FOR RETORTING CURVES.
----------------------------------------------------------------------
LOWER LEVEL COUNT IN FORMAT:  LEVEL-COUNT  LEVEL-COUNT  ETC.
 4= 1

   CHILD REQID SSX002                    LEVEL 03
RECORDS, WHICH ARE KEPT FOR EACH REVIEW CYCLE, MUST BE AUDITED DAILY
BY A MEMBER OF THE PLANT WHO IS LICENSED BY THE FDA SPONSORED SCHOOL
FOR THERMAL PROCESSING OF HERMETICALLY SEALED CONTAINERS(LOW ACID).
THIS PERSON WILL BE RESPONSIBLE FOR PRODUCT SAFETY AND INTEGRITY.
----------------------------------------------------------------------
LOWER LEVEL COUNT IN FORMAT:  LEVEL-COUNT  LEVEL-COUNT  ETC.
 4= 1

   CHILD REQID SSD002                    LEVEL 03
ALL PRODUCTS SHALL RECEIVE A COOL DOWN CYCLE THAT WILL PREVENT
CONDENSATION GATHERING ON CONTAINERS, THEREBY PREVENTING RUST CAUSED
PERFORATION.
----------------------------------------------------------------------
LOWER LEVEL COUNT IN FORMAT:  LEVEL-COUNT  LEVEL-COUNT  ETC.
 4= 2   5= 1

   CHILD REQID SSD003                    LEVEL 03
ANY PRODUCT RECEIVING STRUCTURAL DAMAGE AS A RESULT OF RETORTING
MUST CHANNEL THRU A SALVAGE PROCESS APPROVED BY FDA.  A REPORT
DESCRIBING THIS PROCESS MUST BE ON FILE WITH FDA AND THE CUSTOMER.
----------------------------------------------------------------------

**********************************************************************
```

Figure 10. Example of parent–child report.

so by the large number of requirements. The help provided by ARTS can be divided into three categories.

1. Data base integrity. It is the responsibility of the using project to develop methods of protecting the data base, i.e., restricting authorization to make changes, defining an official version, protecting against data loss caused by hardware failures, etc. When this has been done, the contents of the official version are known to be the current and correct system requirements.
2. Direct computer checks of correctness. Some immediate consistency checks can be run: is the set of requirements with no derivations the same as the set at level 1? Is the test plan number the same for all requirements allocated to a particular module? Getting this information by hand would be time consuming and error-prone. ARTS can make a direct check or tabulate it for easy visual inspection.
3. Aids to the thought process. To evaluate and write requirements, the engineer needs access to a large amount of information. If he stops to gather it up by opening books to the relevant pages, his thoughts have been disrupted. If he can obtain it in seconds with a few ARTS commands, perhaps in a form not previously available, his own creative and analytical abilities can operate at their best.

Figure 11. Requirement definition form.

```
Req. I.D. _________                    Revision __
Status _          Level __  Date ______
Source __________
Derivation _________
Keyword ______________
Summary (50 characters) _____________________
_____________________________________________
_____________________________________________
Responsible Engineer _______________
Phone _________
Text: _______________________________________
_____________________________________________
_____________________________________________
```

B. User Acceptance and Problems

Initial acceptance of ARTS by systems engineering groups at LMSC ranged from enthusiasm to refusal. Greatest acceptance of ARTS came from projects in the early phases of development. Ideally, ARTS would

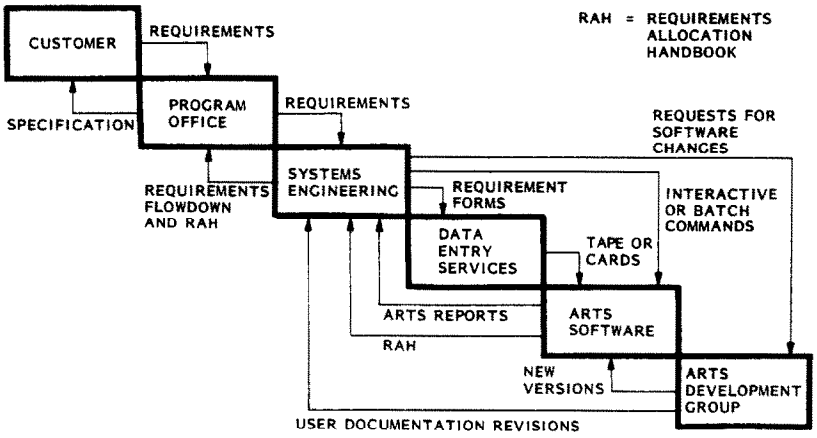


Figure 12. ARTS/systems engineering interfaces.

be introduced at the concept development phase, when only a few top-level requirements have been defined. The number of attributes is also very limited at this stage. The ARTS data base would grow with the project, in number of requirements and in the amount of detail. Using ARTS in this manner provides maximum benefit to a project.

Projects further along in the life cycle can profitably apply ARTS, particularly if their requirements are properly structured and up to date. ARTS is most useful to ongoing projects when major changes (upgrades, extensions, etc.) are planned. These changes usually involve a new set of requirements that replace part of, or are added to, the existing set. The new top-level requirements can be flowed down and the corresponding ARTS data base generated. In this manner, ARTS provides maximum benefit to the development of system changes.

Projects that use ARTS have experienced various problems. The first and most obvious was in obtaining skills to run the program. The development group was required to spend extensive time with all those using ARTS, in what amounted to on-the-job training for the users. The need for documentation and formal training became clear; the results are discussed in Section IIID. It also was concluded that it is not necessary to train all users in all aspects of ARTS. Section IIIC describes various user functions and the amount of training appropriate to each.

Another user problem has included run speed, particularly in the batch input and update modes. Response time for interactive operations has generally been satisfactory.

ARTS is a flexible, general-purpose program. As such, it cannot be expected to meet all user-unique needs in an optimum manner. User meetings have been held to solicit requests for changes to ARTS software. Most such requests have related to increased manipulation capabilities and additional output formats. Within funding and manpower constraints, some requests have been implemented.

C. User Functions

The approximately 60 ARTS system commands available to users range from very simple to exceedingly complex, and their frequency of use varies from many times per day to less than once per month. The individual users also occupy a wide range of experience and familiarity with computer systems. Training all users in all commands and procedures would be difficult and is not necessary.

Most users need to be familiar with only a few basic commands used to generate already-defined reports from an existing data base. This function is mainly interactive and is carried out frequently. It is probably worthwhile to train all systems engineers concerned with requirements to this degree of proficiency.

Training in additional commands and procedures is needed to make additions and changes to a data base. This function is both interactive and batch-oriented. A small number of engineers should be trained in these commands.

Generating new input and output formats requires further training. Experience indicates that one person, the primary ARTS interface for the project, should be trained to this degree of proficiency. This position, referred to as the Requirements Librarian or Data Administrator, is probably a full-time position in the project's Systems Engineering department.

Functions of initial data base design and restructuring are very complex, but occur infrequently. Therefore, LMSC has trained one person, the Applications Specialist, in these functions and made him available to work with using projects. In coordination with the users, he selects attributes, field lengths, and formats;

defines input procedures; and helps define operating procedures. Computer hardware selection may also be involved.

D. Documentation and Training

To avoid extensive involvement by the development group in teaching each new user to run ARTS, a user's manual was written [22]. It contains a general description of the capabilities of ARTS and how it aids systems engineering. Procedures for entering data and obtaining outputs are described. The available commands and explanations of them are given (the explanations are stored as an ARTS data base; they are available to the user on-line as a help operation, and printouts are included in the manual). The manual is not intended to replace the Applications Specialist, but provides a general reference for running the program. While useful as a reference, the manual is limited as an introduction for a new user. Therefore, a training class was generated. Training consists of 9 hr of viewgraph presentation on the ARTS commands and their functions, interspersed with 3 hrs of interactive sessions on the computer terminals, followed by about 3 hr on the applications of ARTS to systems engineering.

Program documentation—a description of the internal construction of ARTS as an aid to program maintenance—is limited at this time.

IV. DEVELOPMENT HISTORY

In January 1980, LMSC undertook the task of acquiring an automated requirements traceability system. Parallel efforts were begun to discuss needed features in a traceability system with potential users at LMSC and to survey commercially available tools.

Discussions were held with the systems engineering and software organizations of several LMSC Space Systems Division projects to determine their needs and desires in requirements traceability. Most were enthusiastic about the prospect of automation. It was soon apparent that two features would have to be provided for maximum usefulness. First, the computer data base must contain attributes other than requirements text, and the user must have the freedom to select the attributes most useful to him. Second, because of the large number of different computer types in use and the classified nature of many projects, the program would have to be available on, or easily converted to, several host computers. Subsequent to these discussions, a preliminary set of requirements for a traceability program was generated and coordinated among the potential users.

Available software was reviewed for systems to perform the entire requirements tracing functions and for data base management systems (DBMS) that might be incorporated into an LMSC requirements tracing program.

A number of existing programs were found that address requirements tracing. Most are, or are part of, methodologies for requirements engineering; these were not considered further because it would not be possible to impose a particular methodology on LMSC project offices. The University of Michgian's PSL/PSA was most nearly suited to our application. It met most of the functional requirements, was usable with almost any development methodology, and was available on many computer types. However, for the particular requirements traceability application, PSL/PSA did not meet all the needs, especially for some specific output formats. It also was intended that traceability be used as a basis for risk and other analysis capabilities. It would have been difficult to learn and modify such a large program.

It was decided that the traceability program should be written at LMSC using a commercially available DBMS if possible. Of the many DBMSs, only a few had the capabilities, flexibility, and portability or potential portability to meet our needs. RIMS was technically suitable, written in FORTRAN IV, easily converted to many computer types, and available to LMSC without charge [23]. Personnel familiar with RIMS were available to aid in conversion and to assist with applications.

Conversion of RIMS to the UNIVAC 1110 and development of additional software for the traceability functions began in March 1980. A prototype installation was generated on the UNIVAC 1110 computer. The prototype was evaluated in cooperation with the Systems Engineering organization of LMSC's National Oceanic Satellite System (NOSS) proposal. The ARTS development group worked closely with NOSS to provide assistance and training, since documentation was very limited. The program performed acceptably, but several needed improvements were identified.

An initial operational version was then generated. The program was written for the DEC VAX-11/780, as well as the UNIVAC. An extensive test plan was undertaken to verify the operational programs. Configuration Management was established, and version release control has been maintained for the operational program.

A preliminary users' manual was generated to accompany the operational program. It consisted of a small amount of introductory material followed by a list of ARTS commands with explanations and examples. Since then, the manual has been expanded so that the introductory material (now on a word processor) includes procedures and extended examples. The com-

mand explanation section is kept current with changes and additions.

ARTS initially was used by two unclassified proposal efforts, NOSS and Solar Electric Propulsion Stage (SEPS). There are currently 14 projects using ARTS, most of them early in full-scale development.

Since the initial operational version was released, various upgrades and improvements have been made. Several new report formats not described in Section IID were developed after the initial release. Conversion to IBM computers was accomplished. Recently the capability was added to read and store titles, paragraphing information, etc., and to print the requirements with this additional information in a specification format.

V. GLOSSARY

Allocation. The process of partitioning or budgeting a performance or functional requirement to the physical subelements of a system. In this manner, the elements of a system that perform specific requirements and that contribute partially to performing others are identified.

ARTS. Automated Requirements Traceability System.

Attribute. A property or description; in the requirements data base, something selected by the user to be stored with the requirements text to provide information about the requirement.

Configuration Management. The function of keeping a record of the actual current contents (of a program, specification, data base, etc.) and of ensuring that these contents are the same as the approved contents.

Derivation. In a hierarchical structure, the derivation is the element(s) at the next higher level that is/are associated with a given element. The lower element is said to be *derived* from the higher element: not to be confused with a derived requirement (q.v.).

Derived Requirement. A requirement at a lower level that is determined to be necessary in order that a top-level requirement can be met. In general, a derived requirement is more specific and is directed toward some subelement.

Flowdown. The process of generating a lower level requirements structure from the (customer-supplied) top level. Consists of decomposition of requirements into more detailed statements at lower levels.

Hierarchy. A type of structure in which each element has a level (1 = highest), and each element is associated with one or more elements at the next higher and lower levels.

Lower-Level Requirement. A requirement that is in the hierarchy below top level or below the level of some given requirement.

Requirement. An authorized and documented need of the customer [9].

System Elements. Those functional groupings of specifiable entities that collectively make up a system, e.g., segment, subsystem, component, unit.

Tree. A hierarchy in which an element at a given level has only one derivation, i.e., flows from one element at the next higher level. Also, a graphical representation of such a hierarchy.

Top-Level Requirement. A requirement that is at level 1 in the hierarchy; it represents a system requirement and has no derivation.

Traceability. Identification and documentation of the derivation path (upward) and allocation/flowdown path (downward) of requirements in the hierarchy. Traceability to and from the design is not explicitly included in this definition.

REFERENCES

1. J. M. Buxton, P. Naur, and B. Randell, (Eds.), Part I: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7–11 October 1968, in *Software Engineering Concepts and Techniques,* Petrocelli/Charter, New York, 1976, pp. 77–78.
2. F. P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering,* Addison-Wesley, Reading, MA, 1975, p. 3.
3. B. C. Deroze, and T. H. Nyman, The software life cycle—A management and technological challenge in the department of defense, *IEEE Transactions on Software Engineering* SE-4, 309–318 (July 1978).
4. M. W. Alford, and J. T. Lawson Software Requirements Engineering Methodology (Development), USAF Rome Air Development Center, Griffiss AFB, NY, RADC-TR-79-168, June 1979, pp. 37–45.
5. W. Myers, The need for software engineering, *Computer* II, 12–26 (Feb 1978).
6. K. A. Hales, Software management lessons learned—The hard way, in *First Computers in Aerospace Conference,* American Institute of Aeronautics and Astronautics, 1977, pp. 182–188.
7. *Software Acquisition Management Guidebook Series,* USAF Electronic Systems Division, Hanscom AFB, MA, many volumes, 1975 and later.
8. *Structured Programming Series,* USAF Rome Air Development Center, Griffiss AFB, NY, 1975.
9. SSD Standards and Practices for Software Engineering, Lockheed Missiles & Space Company, Inc., Sunnyvale, CA, Document Code 63, May 1983.
10. M. W. Alford, and J. T. Lawson, Software Requirements Engineering Methodology (Development), USAF

Rome Air Development Center, Griffiss AFB, NY, RADC-TR-79-168, June 1979, p. 37.
11. W. W. Royce, Software requirements analysis, in *Practical Strategies for Developing Large Software Systems* (E. Horowitz, ed), Addison-Wesley, Reading, MA 1975, p. 59.
12. B. W. Boehm, Software engineering, *IEEE Transactions on Computers* C-25, 1226–1241 (Dec 1976).
13. R. L. Glass, Persistent software errors, *IEEE Transactions on Software Engineering* SE-7, 162–169 (March 1981).
14. M. W. Alford, and J. T. Lawson, Software Requirements Engineering Methodology (Development) USAF Rome Air Development Center, Griffiss AFB, NY, RADC-TR-79-168, June 1979, p 38.
15. E. D. Nelsen, Solving the requirements allocation mystery, Lockheed Missiles & Space Company, Inc., Sunnyvale, CA, Internal Report, Feb 1980.
16. M. W. Alford, and J. T. Lawson, Software Requirements Engineering Methodology (Development), USAF Rome Air Development Center, Griffiss AFB, NY, RADC-TR-79-168, June 1979, pp. 44–45.
17. W. W. Royce, Software requirements analysis, in *Practical Strategies for Developing Large Software Systems* (E. Horowitz, ed.), Addison-Wesley, Reading, MA, 1975, p. 62.
18. W. W. Royce, Software requirements analysis, in *Practical Strategies for Developing Large Software Systems* (E. Horowitz, ed.), Addison, Wesley, Reading, MA, 1975, p. 69.
19. M. W. Alford, and J. T. Lawson, Software Requirements Engineering Methodology (Development), USAF Rome Air Development Center, Griffiss AFB, NY, RADC-TR-79-168, June 1979, p. 99.
20. D. J. Paul, Arts and Systems Engineering, Lockheed Missiles & Space Company, Inc., Sunnyvale, CA, Internal Report, April 1981.
21. D. J. Paul, NOSS Requirements Allocation Handbook, Lockheed Missiles & Space Company, Inc., Sunnyvale, CA, LMSC/D762163, Feb 1981.
22. T. C. Burke, Automated Requirements Traceability System (ARTS) users manual, Lockheed Missiles & Space Company, Inc., Sunnyvale, CA, LMSC/D794416, April 1981.
23. RIMS Design Specification, NASA-Johnson Space Center, Houston, TX, LEC-9564, Feb 1976.