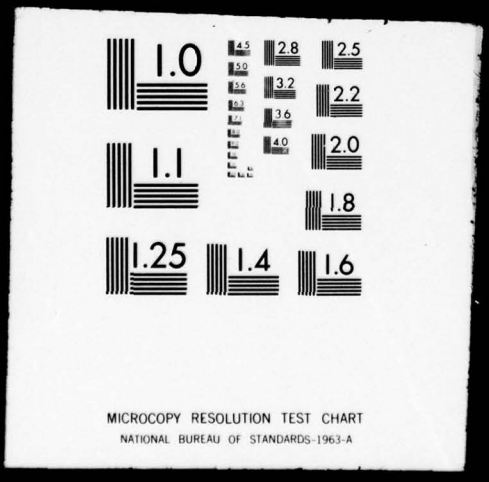
AD-A061 009 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
CADSAT APPLICATION GUIDEBOOK.(U)
AUG 78
F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-125 NL



ESD-TR-78-125

LEVEL II

CADSAT APPLICATION GUIDEBOOK

ISDOS Project
University of Michigan
Department of Industrial and Operations Engineering
Ann Arbor, Michigan 48109

August 1978

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

This Technical Report has been reviewed and is approved for publication.

ANN MELANSON
Project Engineer

CHARLES J. GREWE, Jr., Lt Col, USAF
Chief, Technology Applications Division

FOR THE COMMANDER

ERIC B. NELSON, Colonel, USAF
Acting Director, Computer Systems Engineering
Deputy for Technical Operations

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ESD-TR-78-125	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): CADSAT APPLICATION GUIDEBOOK,	5. TYPE OF REPORT & PERIOD COVERED: Manual Rept. for July 1976 to March 1977.
6. PERFORMING ORG. REPORT NUMBER: CDRL Item 023	
7. AUTHOR(s): ISDOS Project	8. CONTRACT OR GRANT NUMBER(s): F19628-76-C-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Michigan, Department of Industrial & Operations Engineering, Ann Arbor, MI 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE 64740F, Project 2237
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Technical Operations, Electronic Systems Division, Hanscom AFB, MA 01731	12. REPORT DATE: August 1978
	13. NUMBER OF PAGES: 94
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

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

18. SUPPLEMENTARY NOTES

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

Automated Tools
Computer-Aided Design
Information Processing
Information System Requirements
Requirements Analysis
Requirements Language
Requirements Specification
Specification Analysis

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

This guidebook is intended to provide general information to users of the Computer-Aided Design and Specification Analysis Tool (CADSAT). The guidebook explains how users can make the most effective use of the User Requirements Language (URL) and the User Requirements Analyzer (URA). This report is primarily concerned with the use of CADSAT in describing the requirements of a proposed system such as an information-processing system and in investigating and analyzing the completeness and consistency of that system.

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

403 871

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

78 10 23 047

LIST OF TABLES

Notation

CADSAT reserved words are in capital letters, e.g., ATTRIBUTES.

CADSAT user names are in lower case when given in URL statements e.g., INPUT time-card; enclosed in single quotes when used in the text, e.g., 'time-card'.

1. INTRODUCTION

The Computer-Aided Design and Specification Analysis Tool (CADSAT) System is designed to be used by people such as systems analysts, target system users, project leaders, systems designers and programmers, that have various viewpoints with respect to the target information processing system. This guidebook, is intended to provide individualized information to the various groups of CADSAT users on how they can make the most effective use of User Requirements Language (URL) and User Requirements Analyzer (URA).

This guidebook is not intended to replace or overlap with other CADSAT Documentation. Furthermore, it should not be read as the first introduction to CADSAT.

The reader of this guidebook is expected to have attended a training course in URL/URA, thus to know the language and related software, and to have had some experience in system analysis.

The life cycle of an information processing system may be considered to consist of the following stages:

a. Formulation of the request for a new system[1] or modification of an existing one expressed in the form of user requirements with respect to the target system.

b. User's requirements analysis, object[2] system investigation, present system investigation, logical design and impact analysis of the target system.

c. Physical design of the target system.

d. System construction and component testing.

e. System test, conversion and implementation.

f. Operation and maintenance of the target system.

This guidebook is primarily concerned with the use of the CADSAT system in stage b for requirements analysis, object system investigation, present system investigation and logical design of the target system. There are a number of basic activities which must be accomplished during that stage. However, they are performed in different ways and in a different sequence by organizations. Therefore, this guidebook consists of a collection of independent topics rather loosely related but potentially useful during that stage. This reduces the

---

[1] System if not used with other modifier, stands for information processing system.

[2] Object system - any business, governmental, adminstrative or other organization which the target information processing system is intended to serve.

probability of conflicts with the methods presently applied in organizations. The maximum use of the guidebook can be expected immediately by system analysts, project leaders, systems department and, in long range by the users of the target system.

At Electronic System Division (ESD) the target system users are the ccmmands for whom the system is being built such as ADCOM, SAC, MAC, etc.

The primary users of URL/URA will be the systems analysts. They will be required to become thoroughly acquainted with all the language options, analyzer commands and various URA reports in relation to the ESD standards and conventions. At ESD, the system analysts are either the System Program Office (SPO) personnel or contractors hired to analyze system requirements and produce a system specification.

Project leaders provide the immediate supervision of the analysts. They will be particularly interested in using the URA for the task assignment of project team members, monitoring of progress and performance, checking for completeness, consistency and correctness and for the final systems documentation at different stages of the systems life cycle. At ESD, project leaders are the System Program Offics (SPO) directors.

Although at ESD the physical design of the system, programming and testing are done by contractors, under the supervision and control of the System Program Office, those people need sufficient understanding of URA outputs in order to be able to transform the requirements stored in URA data base into an operational system.

However, the decisive factor in correct, efficient and effective user of CADSAT is the systems department. In this case it is the Headquarters Electronic System Division itself. Therefore a separate section 2 of the guidebook is intended mainly for the management of the systems department.

The headings of the guidebook are supplemented, where necessary, with notes indicating to whom they are intended. These are provided to reduce the amount of information a CADSAT user must read which will not be of direct use to him.

2. THE SYSTEMS DEPARTMENT AND CADSAT SYSTEM

(for managers of systems departments)

In practice, URL/URA will probably be used initially on a selected small project. Eventually, however, it will be most effective when it is adopted as a standard system development tool. It is, therefore, desirable to plan for the adoption of URL/URA in the whole Systems Department.

Systems Departments usually perform the following functions:

- management of the development, operation and maintenance of information processing systems,
- development of new systems,
- operation of existing systems,
- maintenance of existing systems,
- staff operations.

The need for improvements exist in all of them.

Most organizations manually perform all activities related to the determination of requirements or specifications for a proposed information processing system. Using the present system as a starting point, analysts collect data about requirements for the new system, primarily through a process of interviewing user department personnel. This data is then collated, analyzed and summarized to produce a set of specifications for a new system. These specifications, together with an implementation plan and cost/benefit analysis, are normally included in a document having a name similar to one of the following: Feasibility Report, System Definition Report, System Specification Report, or Functional Requirements Report.

The Computer-Aided Design and Specification Analysis Tool system (CADSAT) supports the manual procedures by providing the capability to use the computer for some of the clerical activities, and by providing better analysis of the requirements than is possible with manual methods. The data is still collected from the usual sources by the analysts. However, as it is collected, it is expressed in a formal language called the User Requirements Language (URL). The data expressed in URL is entered into a computerized data base, either incrementally as it is obtained, or in batches as desired. As it is entered, the User Requirements Analyzer (URA) checks for correctness and consistency of the new data with that already in the data base.

At any time, URA can be asked to produce reports on all or any selected part of the data in the data base. The data used in any given report may have been entered by different analysts at different times. During the production of a report, URA carries out numerous checks and analyses and produces warnings and diagnostics as appropriate.

When the project is complete, the final specification document required by the organization can be produced automatically.

2.1 Present Manually Performed Activities

That process in general can be considered to consist of five major types of activities (Figure 2.1):

1) Data collection. Information about the data flow in the present system, requirements for new information, and the potential new system organization is collected and recorded.

2) Analysis. The data that has been collected is summarized and analyzed. Errors, omissions and ambiguities are identified and corrected. Redundancies are identified. The results are prepared for review by appropriate groups.

3) Design of a proposed system. Processes to be performed by the system are selected. Alternatives for a new system or modifications of the present system are developed and examined. The "new" system is developed and described.

4) Evaluation. The benefits and costs of the proposed system are determined to a suitable level of accuracy. The operational and functional feasibility of the system are examined and evaluated.

5) Improvements. Usually as the result of the evaluation, a number of deficiencies in the proposed system will be discovered. Alternatives for improvement are identified and evaluated until further possible improvements are not judged to be worth additional effort. If major changes are made, the evaluation step may be repeated; further data collection and analysis may also be necessary.

In practice the type of activities outlined above may not be clearly distinguished and may be carried out in parallel or interactively with increasing levels of detail. Throughout the process, however it is carried out, results are recorded and documented. Documentation includes narrative statements, lists, tables, arrays and charts of various types. Several techniques, procedures, and analysis methods have been developed to aid in the process. Most of these are manual methods though some computer-aided methods have also been developed.

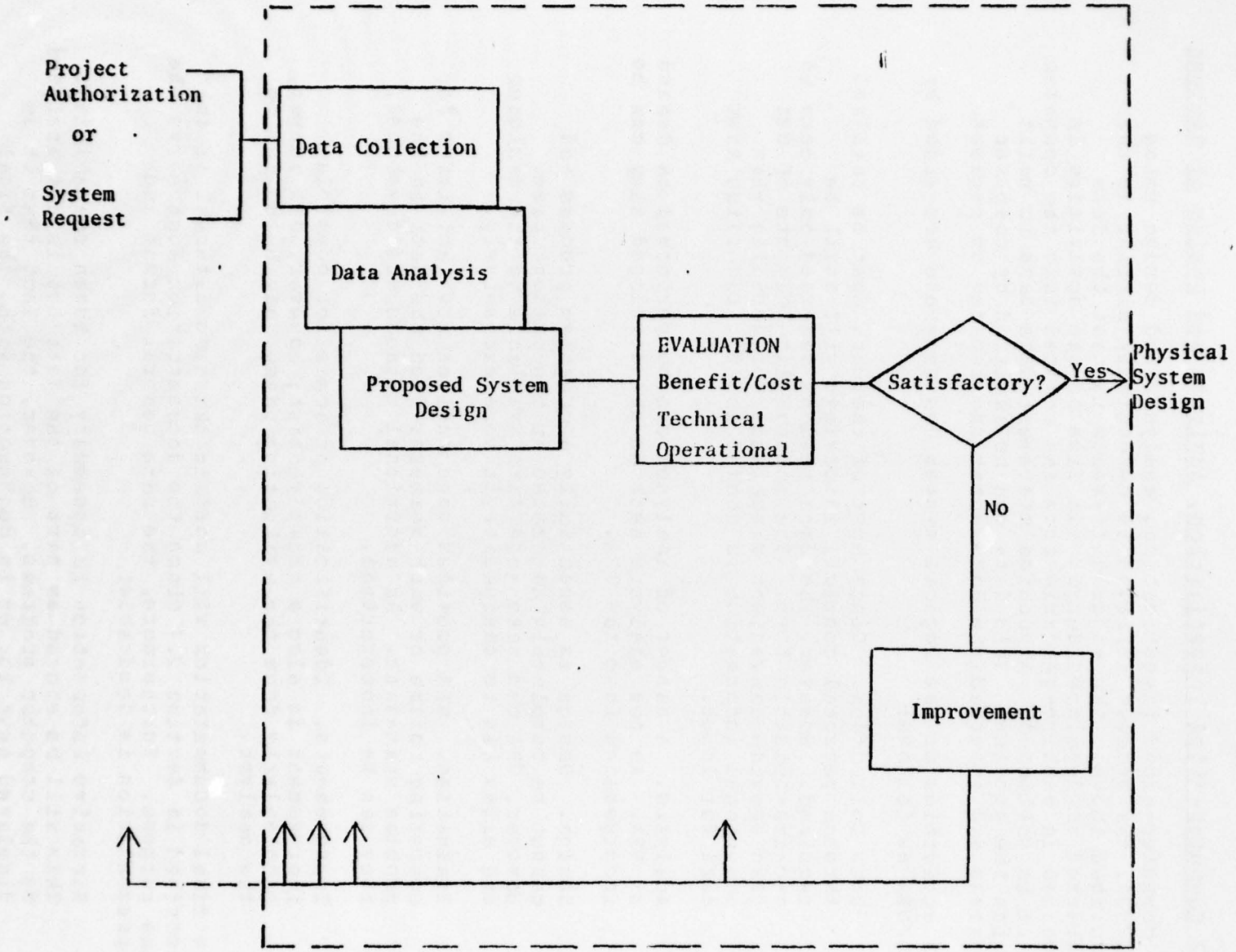


Figure 2.1 Logical System Design

2.2 Computer-Aided Investigation, Analysis and Design of Systems

In computer-aided investigation, analysis and design using URL/URA, the process followed is essentially similar to that described above. The major difference is that the data collected or developed during all five of the activities is recorded in machine-readable form and entered into the computer as it is collected. A problem statement data base is built during the process. This data can be analyzed by computer programs and intermediate documentation prepared on request.

The activities of the logical system design phase are aided by URL/URA as follows:

1) Data Collection. Since most of the data must be obtained through personal contact, interviews will still be required. However, the data need be recorded only once in machine-readable form. The intermediate outputs of URA also provide convenient checklists for deciding what additional information is needed and for recording that data for input.

2) Analysis. A number of analyses can be performed on demand by URA. As new analysis methods are developed they can be incorporated into the URA.

3) Design. Design is essentially a creative process and cannot be completely automated at the present time. However, URA can make more data available to the designer and allow him to manipulate it more extensively.

4) Evaluation. URA provides some rudimentary facilities for computing volume or work measures from the data in the problem statement. As additional methods are developed, they can be incorporated.

5) Improvements. Identification of areas for possible improvement is also a creative task; however, URA output, particularly from the evaluation phase, may be useful to the analyst.

The final documentation will contain the same material as that described in Section 2.1 since the documentation must serve the same purpose. Furthermore, the same general format and representation is desirable:

1) Narrative information is necessary for human readability. This will be stored as part of the data but is not analyzed by the computer programs. However, the fact that it is displayed next to, or in conjunction with, the final description improves the ability of the analyst to detect discrepancies and inconsistencies.

2) Lists. Since lists are prepared from the data base, they are up-to-date and can be more easily rearranged in any desired order.

3) Tables, arrays, matrices. These are also prepared automatically and hence, are up-to-date.

4) Diagrams and flow charts. The URA system produces graphical outputs which display the relationships between objects.

The basic objective of the CADSAT project is to provide an integrated and complete set of computer-aided methods to aid first in the documentation of the description of a proposed information processing system, and second in the process leading to that description.

2.3 Project Organization

The project organization appropriate to a particular project will depend on the size of that project (for example, the number of analysts involved) and, to a lesser extent, on the type of application. In general, where several analysts are working on the same system, it is probably wise for a single person to have responsibility for locating and correcting inconsistencies between requirements stated by various analysts, and for finding indications that more data are needed. Depending on the size of the project, this person may be one of the group of analysts, the project leader, or an analyst assigned to only the coordination function. The use of CADSAT should reduce the amount of direct interaction between the analysts. Rather than directing questions to each other periodically, they may simply refer to the data base for information on other portions of the system.

Both the management of Systems Department and all project leaders can be supported by CADSAT in performing the following management activities:

- keeping record of work assignments given to the project teams and project team members,
- assuring recorded intercommunication links between the Systems Department Management, project leaders, and project team members working even in remote sites of the object organization,
- monitoring the overall work in progress,
- determining and observing the systems department standards concerning the content and form of the final documentation generated for the internal and external use,
- monitoring the progress of projects against budget and time schedule.

The individual project leaders can be very effectively assisted by CADSAT in:

- detecting incompleteness and inconsistencies between the URL statements made by various team members,
- monitoring the individual performance of team members.

However, in order to obtain full benefits of CADSAT assistance in the management of projects and systems maintenance, some standards for the use of KEYWORDS, ATTRIBUTES, ATTRIBUTE-VALUES, SOURCES and PROBLEM-DEFINERS have to be established before the CADSAT system is used.

In order to use the CADSAT System, an organization must have access to the URA software and must provide for data entry facilities and the integrity of URA data bases.

Beyond this, a suitable environment must be maintained in order to ensure successful use.

The proper support must be provided. This includes:

- an in-house enthusiast who can solve problems that arise and provide in-house consulting,
- education and training,
- documentation, guidebooks, standards, etc.

An obvious requirement, of course, is that the change in methodology must have the full support of management.

It must be stressed that URL/URA is only a tool. The potential benefits can be obtained only if the tool is used correctly. This requires close management supervision of the complete system development procedure being followed by the organization.

2.4 Interfacing with Presently Applied Methods and Procedures

Each systems department uses some standards, techniques, and procedures. It also has some personnel with various degrees of training, experience and skills.

The present methods used in defining the system specifications or requirements need not change with the introduction of CADSAT. URL can accommodate a wide variety of methods, and, where a well-established and useful approach exists, it is least disturbing to the organization to preserve that approach. On the other hand, if there is no overall method in use, one should be selected and its use encouraged.

2.5 Resources Required for CADSAT Use

1) Input Facilities

URA statements may be input in batch mode. If this is done, keypunch facilities will be required. Analysts may write URL on forms, such as those found in Appendix G to the URL User's Manual. Cards may be punched directly from the forms and the resulting URL used as input to URA in batch mode.

Alternatively, URL statements may be input from a terminal. The analyst or an operator, may enter URL into a file using the locally available text editor, and may interactively invoke URA to input the file into the URA data base.

Another approach is a combination of the two techniques. Large quantities of input may be handled in batch, using the services of a keypunch group. Small additions and corrections to the data base may be made by the analyst himself, working interactively from a terminal.

2) Storage for Data Bases and Input Files

Sufficient storage must be provided for any URA data base to be used, along with additional storage space for each analyst's input files and temporary files for output as needed.

3) Documentation

A complete set of CADSAT documentation should be available. Each analyst should have his own copies of the URL and URA User's Manual, as he will be likely to refer to them frequently.

4) Consulting Help

While it is possible to learn to use URL and URA by making good use of the documentation, the project is likely to proceed more smoothly if an experienced user of CADSAT is available for consultation. If the project leader is an experienced user, he will be able to establish and encourage the use of reasonable conventions for the URL written by his analysts.

5) Data Base Security

The data base must be protected from unauthorized or unintentional modification or destruction. This is normally done by proper use of the access or file permission facility of the operating system. In addition, if there is no regular, frequent file save performed by the operations staff, the data base should be saved periodically by a member of the project team.

6) Output Facilities

URA output may be obtained in either batch or interactive mode. Large outputs are best produced in batch mode. Small reports to answer questions occurring to the analyst may be more useful when obtained immediately.

2.6 Selection of Projects

Projects should be properly chosen for CADSAT use. Management should not attempt to use CADSAT to help a project that is falling behind to meet its deadlines. Adequate training in CADSAT use requires a certain amount of time, so the proper training is not likely to be done in a project that is behind schedule. Thus CADSAT is likely only to add to the difficulties, and is not likely to be seen as a useful tool.

CADSAT use is best initiated with a project that has not yet begun the requirements analysis phase. In this case, time will be allowed for training analysts in its use. The systems analysis methods that are utilized will receive any adaptations that seem desirable, and the effort is more likely to be properly coordinated for this kind of project.

This is not to say that CADSAT would not be beneficial if introduced after the requirements analysis phase has started. The information that has already been collected can still be expressed in URL, possibly by a single analyst working independently. URA can still be used to generate reports to evaluate the consistency and completeness of the data base. At this point, however, a large number of errors and omissions are likely to be discovered, which had not been detected by manual analysis techniques. Thus, careful coordination between the analysts must take place to correct the errors, which may be more difficult to handle due to their volume in later stages of system development and numerous interrelationships.

It is not necessary that CADSAT be used for all projects within an organization. While this is certainly feasible, there are benefits to a pilot project approach. The use of CADSAT could begin with one or a few projects within an organization. Upon completion of these projects, an evaluation should consider questions such as the following:

1) Were the facilities provided for CADSAT use adequate, or was there a need for more terminals, more storage space, etc.?

2) Do any changes in the mechanics of CADSAT seem advisable? Should large batches of URL input be keypunched and entered in batch, rather than being input by the analyst interactively? Should some reports that have been produced interactively be produced in batch mode, or vice versa?

3) Is the organization's method for requirements analysis compatible with the CADSAT environment? Could a change in

this method produce better results?

4) If URA was not used directly by designers, might it have been effectively used that way? If it was used, were the results different from what might have been expected from previous methods?

The answer to these questions should be valuable in providing a smooth transition to the CADSAT environment for additional analysts and system development projects.

2.7 Benefit/Cost Analysis of CADSAT Application

The major benefits claimed for computer-aided documentation are that the "quality" of the documentation is improved and that the cost of design, implementation and maintenance will be reduced. The "quality" of the documentation, measured in terms of preciseness, consistency and completeness is increased because the analysts must be more precise. The CADSAT software performs the checking, and the output reports can be reviewed for remaining ambiguities, inconsistencies and omissions. While completeness can never be fully guaranteed, one important feature of the computer-aided method is that all the documentation that "exists" is in the data base, and, therefore the gaps and omissions are more obvious. Consequently, the organization knows what data it has, and does not have to depend on individuals who may not be available when a specific item of data about a system is needed. Any analysis performed and reports produced are up-to-date as of the time it is performed. The coordination among analysts is greatly simplified since each can work in his own area and still have the system specifications consistent.

Development will take less time and less cost because errors, which usually are not discovered until programming or testing, have been minimized. It is well recognized that one reason for the high cost of systems development is the fact that errors, inconsistencies and omissions in specifications are frequently not detected until later stages of development: in design, programming, systems test or even operation. The use of URL/URA during the specification stage reduces the number of errors which will have to be corrected later. Maintenance costs are considerably reduced because the effect of a proposed change can easily be isolated thereby reducing the probability that one correction will cause other errors.

The cost of using a computer-aided method during logical system design must be compared with the cost of performing the operations manually. In practice the cost of the various analyst functions of interviewing, recording, analyzing, etc. are not recorded separately. However, it can be argued that direct cost of documenting specifications for a proposed system using URL/URA should be approximately equal to the cost of producing the documentation manually. The cost of typing manual documentation is roughly equal to the cost of entering URL statements into the computer. The computer cost of using

URA should not be more than the cost of analyst time in carrying out the analyses manually. (Computer costs, however, are much more visible than analysts costs.) Even though the total cost of logical system design is not reduced by using computer-aided methods, the elapsed time should be reduced because the computer can perform clerical tasks in a shorter time than analysts.

2.8 Impact on the Systems Department

URL/URA, if used properly, should improve the productivity of systems analysts by replacing clerical tasks in manual documentation methods with computer-aided facilities. It is not intended to replace analysts or reduce the number required in an organization. It is also not intended to enable untrained or unqualified analysts to do systems analysis. It should permit qualified analysts, adequately trained in URL/URA and with the right computing facilities and management support, to improve the quality of their work.

3. EXPRESSING THE PROJECT TASK IN URL

(for systems analysts)

An information processing system usually originates with the perception of a need. In many cases, this comes from the people who will make use of the system. It may, however, come from a service group, which perceived a common need of several units within an organization. In either case, the initial user's request or proposal for a new system should contain the same basic information about the target system.

After a user's request or an offer of a service group has been approved, it becomes the project task or contract statement of work. Since everything that will be conceived, designed, implemented and put into operation during the target system's life cycle should be compared and evaluated against the project task, the information both on the former and the latter must be expressed in URL, if CADSAT is intended to be used throughout the next stage of the system life cycle. Otherwise, a meaningful computer assisted analysis of user requirements would not be feasible.

3.1 Basic Principals

The project task should consist of the following parts:

1) The subject of the request or task.

2) The main objective of the user which is to be achieved by means of the implementation and operation of a system, expressed in terms of an effect measurable outside of the target system.

3) The main function of the target system defined by one verb with only indispensible modifiers limiting the type of activities the user expects the system to perform. These should be expressed without any suggestions to their quality and how they should be done in order not to constrain the inventive creativity of the project team.

4) Definition of requirements and constraints, imposed by the user on the solution of the problem, which concern the target system's outputs, applied methods of processing, inputs, used facilities, staff, and operational environment. In addition, each of these factors should be considered in three aspects:

   - Its type, quality, form, and time requirements,
   - Its size (amount per time or distribution in time),
   - Its changes over time, seen from the point of user's needs external to the target system.

   All requirements and constraints must be placed into one of two categories - mandatory and optional. Mandatory means

that the project will be abandoned or rejected if any one of this kind of requirement could not be met. For each optional requirement the user is obliged to indicate the upper value of the worth he assigns to it. It should be expressed in terms of additional cost or decreased amount of main effect that he is willing to tolerate, if it is met. The user should be aware that each additional requirement or constraint imposed on the solution never improves it, but usually deteriorates it in respect to cost and performance. Therefore, he should specify only those requirements that are indispensable and worthy, considering only the outside of the target system and trying to limit their number and improve their clarity.

5) The definition of the boundaries of the target system in respect to:

   - time (earliest date on which full operational capabilities are supposed to be achieved and the minimum period of its operation without major alternations) and,

   space (for each input and output mentioned in previous parts - where it originated, and where it is destined to fulfill the requirements of the target system.)

6) A synthesized evaluation criterion of the target system's project and operation in the form of a function relating the value of the system to the accomplishment of its main objective and fulfillment of all requirements and constraints, according to the user's scale of preferences.

7) Optional suggestions on subdivision of the target system into subsystems, in cases where external organizational reasons, as seen from the user's point of view make such subdivisions indispensible. But then the user is obliged to specify all the above parts of his requirements for each subsystem, he singled out.

3.2 Example of a Project Request in Plain English

A user's request for a modification of a payroll system will be used as a simple example. Assume it has been formulated by a very demanding user. He is not going to pay for anything which does not comply with his mandatory requirements. He is fully aware of his ojectives and knows exactly what he wants to achieve. He is a user who possesses the necessary managerial and systems engineering skills to formulate them clearly but does not want to unwillingly constrain the freedom of the project team in their search for effective solutions. He states only what matters to his business outside of the information processing system and refrains from saying anything about the internal solution of the system, since he has no organizational reason to do so.

"The subject of the following request is the improvement of a payroll system which is in operation at Organization X. The

main business objective is to reduce the present overall operation costs. Therefore, the main kinds of improvements which can justify any computerization project in that field is the reduction of the weekly operation costs of payroll system from the present level of $1.00 per employee. In order to be approved this project must be competitive to other computerization projects currently under study (see objective function).

The main function of the payroll system is to derive the payroll outputs from the payroll inputs. The size of the main function is measured by the number of employees for whom it is performed.

The following requirements and constraints are imposed on the solution of the problem. If not otherwise explicitly stated, all of them are considered to be mandatory. That means that the project will be abandoned or rejected if any one of these requirements could not be met.

The mandatory outputs to be produced each week are: paychecks for all elegible employees (within the range from 15,000 to 20,000), payroll report, hired-employee-report and terminated-employee-report for the payroll-department. All these outputs must contain the content of the present reports. The payroll report may optionally be enhanced by an automatically produced pay-exception-messages presently produced manually at the cost of not more than $.10 per message.

The payroll processing must update the payroll master information weekly. The programmed procedure must comply with present federal, state, company and union contractual regulations.

The payroll system is constrained to utilize the company's computer facility. The investment expenses of the project will be augmented by $3000 per discharged employee. The present company's cost of capital is at least 15%.

The time and space boundaries of the payroll system are defined as follows: The system is due to be implemented by the end of the current year and is expected to be in operation for at least five years. The payroll inputs have to be delivered to the payroll department by all company's organizational units not later than 4 p. m. on the last work-day of each week. The payroll outputs should be delivered at the risk and expense of the payroll system to the receiving departments or mailed to the employees, if requested.

The present value of the investment involved with the project will be considered as the combined evaluation criterion of the project. The value of the reduction of the operation costs of the payroll system should be computed for the present (15,000), maximum (20,000) and average number of employees according to the following formula:

$R = E(Cp-Cr) + 0.1 + Pe$ [dollars per week] where:

R - reduction of the operation costs of the payroll system, the project is going to produce in dollars per week,

E - number of employees,

Cp - present operation costs of the payroll system in dollars per week and employee(without derivation of the pay exceptions),

Ct - total operation costs of the target payroll system in dollars per week and employee,

Pe - number of pay exceptions automatically produced per week,

0.1 - cost of manual derivation of pay exceptions in dollars per message.

No subdivision of the payroll system is necessary from the organizational point of view.

3.3 The Same Example Stated in User Requirements Language (URL)

The example of the project task statement described above can be expressed in a set of URL statements. Subsequently, most of them can be automatically enhanced by statements obtained from interviews, which can be analyzed for consistency with the previously stored statements. Also most of the final logical specification of the system can be thoroughly analyzed with the assistence of the computer for compliance with the project task stored in the URA data base. Where the computer is not yet feasible, the statements can be automatically presented to the analyst for manual checks.

The method presented for the project task description in URL is based on the following assumptions and rules:

a) The task payroll system description should be stored in the same data base but in clear distinction to the present payroll system and the target payroll system description. The latter will be the outcome of the project team's efforts and can be considered at the highest level of abstraction as a PROCESSOR which PERFORMS the necessary PROCESS payroll-processing, which in turn RECEIVES some payroll-inputs and GENERATES the payroll-outputs for users specified in the project task.

b) All system objects mentioned in the project task should be described only to the extent that the constraints and requirements specified therein concern them.

c) As far as possible, all user requirements should be expressed in standard URL relations. URL comment entries useful for the documentation generator should also be expressed in computer analyzable form. The comment entries later can be checked manually by the analyst to determine whether they comply with other statements about the same

objects.

d) The common SOURCE of all the following URL statements is the user's request.

e) The naming convention must provide for a clear distinction of all objects about which the project task contains any statements significant for the final specification of the target system. Here the adjective "task", where possible, or its abbreviation "t" will be used throughout the example as the leading term in all analyst defined names.

Later, this naming convention makes feasible automatic checking by the URA (when combining the object names used in project's task description with the same semantic names finally used by the project team) of whether the final statements made by the project team are not contradicting the statements which represent the user's requirements in the project task. It also ensures that they do not later interfere with similar statements put into the data base by members of the project team.

All further explanatory comments will be enclosed in /*...*/ according to the URL convention in order to provide the necessary explanations without hindering their use as computer input for URA.

```
/*Project Task for Payroll System expressed in URL for
documentation purposes and for computer assisted analysis where
possible*/

/*1.1 Subject of the Request                                    */
MEMO subject-of-the-request; DESCRIPTION; /*for documentation
generator*/
Improvement of the payroll system in operation at Organization
X;

PROCESSOR task-payroll-system; ATTR: for-organization x,
to-be-an-improved-version-of payroll-system-in-operation;
SYNONYM IS: tprs; SEE-MEMO subject-of-the-request;
/*Major components of the previous DESCRIPTION statement
expressed in URA analyzable form*/

/* 1.2 Main Objective of the Task                               */

MEMO task-main-objective; DESCRIPTION;
Mandatory reduction of the weekly payroll operation costs of the
present payroll system from the level of $1.00 per employee in a
competitive way with other computerization projects.  (See
section: Objective Function);

APPLIES TO task-payroll-system; /* for documentation generator
and manual analysis, whether the consumption rate of operation
costs later declared by the project team (after the feasibility
study) meets the user's main objective */

/* 1.3 Main Function of the System                              */
```

```
MEMO task-system-main-function; DESCRIPTION; /*for documentation
generator*/

To derive the payroll outputs from payroll inputs.  (The
intensity of the main function is measured by the number of
legible employees for whom it is performed);

APPLIES TO task-payroll-system;
PROCESS task-payroll-processing; DERIVES task-payroll-outputs
USING payroll-inputs; PERFORMED BY task-payroll-system;
ATTRIBUTE IS: t-measure-of-intensity t-number-of-employees;

/*(description of the system's main function in a form
analyzable by URA)                                          */

/* 1.4 Design Requirements and Constraints
(imposed on system's outputs, processes, inputs and means)
                                                            */

OUTPUT task-payroll-outputs; GENERATED BY payroll-processing;
RECEIVED BY t-departments-and-employees; HAPPENS 1 TIME-PER
week;

/*must have*/ SUBPARTS: t-pay-checks, t-payroll-report,
t-hired-employee-report, t-terminated-employee-report;

OUTPUT t-pay-checks; RECEIVED BY t-employees; CONSISTS OF
t-number-of-employees-syspar t-present-pay-check-content;

DEFINE t-number-of-employees-syspar AS SYSTEM-PARAMETER; VALUES
ARE 15000 THRU 20000;

OUTPUT t-payroll-report; RECEIVED BY t-payroll-department;
CONSISTS OF t-present-payroll-report-cont;
t-optional-exception-messages;

GROUP t-optional-exception-messages; KEYWORD t-optional;
CONSISTS OF t-number-of-pay-exceptions
t-present-exception-content; ATTRIBUTE IS:
t-increases-objective-value-by ten-cents-per-message;

OUTPUT t-hired-employee-report; RECEIVED BY
t-payroll-department; CONSISTS OF
t-present-content-on-hired-empl;

PROCESS task-payroll-processing; UPDATES
t-payroll-master-information; HAPPENS 1 TIME-PER week;
PROCEDURE;

In compliance with present federal, state, company and union
contractual regulations;

TERMINATION-CAUSES operational-objectives-accompl;

INPUTS payroll-inputs; RECEIVED BY task-payroll-processing;

/*Additional requirements and constrains concerning the use of
```

```
means*/

PROCESSOR task-payroll-system; SUBPART IS:
t-companys-computer-facility;
/*t - stands for mandatory component */ CONSUMES
t-operation-costs-per-year AT RATE OF t-fifteen-percent PER
task-investment; CONSUMES task-capital AT RATE OF
t-three-thousand-dollars PER t-discharged-employee;

/*1.5.1 System Time Bounderies                                    */

PROCESSOR task-payroll-system; ATTRIBUTES ARE:
t-due-to-be-implemented t-till-the-end-of-1977,
t-minimum-operational-period t-five-years;

/*1.5.2 System Space Boundaries                                   */

INPUTS task-payroll-inputs; ATTRIBUTES ARE:
t-delivered-weekly-till t-last-work-day-4pm,
t-delivered-weekly-to t-payroll-department-location;

OUTPUTS task-payroll-outputs; ATTRIBUTE ARE:
t-delivered-weekly-to t-receiving-orgu-locations;

OUTPUTS t-paychecks; ATTRIBUTES ARE: t-delivered-weekly-to
the-employing-orgunits, t-mailed t-if-requested;

/*1.6 Objective Function                                          */

MEMO task-objective-function; DESCRIPTION;

The present value of the investment involved with the project
will be considered as the combined evaluation criterion of the
project.  The value of the reduction of the operation costs of
the payroll system should be computed for the present (15,000),
maximum (20,000) and average number of employees according to
the following formula:

     R = E(Cp-Ct) * 0.1 * Pe [dollars per week] where:

R - reduction of the operation costs of the payroll system, the
     project is going to produce in dollars per week,
E - number of employees,
Cp - present operation costs of the payroll sytem in dollars per
     week and employee (without derivation of the pay
exceptions),
Ct - total operation costs of the target payroll system in
     dollars per week and employee,
Pe - number of pay exceptions automatically produced per week,
0.1 - cost of manual derivation of pay exceptions in dollars per
     message;

APPLIES TO: task-payroll-system;

/*1.7 Optional Subdivision                                        */

MEMO task-subdivision; DESCRIPTION;
```

No subdivision of the task-payroll-system is necessary from the organizational point of view;

APPLIES TO; task-payroll-system;

/*1.8 Other General Statements */

DEFINE user-request AS SOURCE, none AS SECURITY; APPLIES TO:

subject-of-the-request, task-payroll-system,
task-main-objective, task-system-main-function,
task-objective-function, task-payroll-outputs, t-paychecks,
t-number-of-employees, t-payroll-report,
t-optional-exception-messages, task-payroll-processing,
t-hired-employee-report, t-terminated-employee-report,
task-subdivision;

DESCRIPTION;

If not otherwise explicitly stated by the adjective "optional" used as the first term of the analyst defined names, all URL statements which refer to the SOURCE user-request are considered as mandatory. It means, that the project will be abandoned or rejected if any of this kind of requirements could not be met by the project team;

/*In this case, the common SOURCE for all previous URL statements is the user's request. Similarly no security requirements are declared
- The End of Project Task Expressed in URL - */

Of course, in real situations many users are not able to formulate their requests in a rigid way. But then, the analyst should know what is missing, and what in addition he should ask for during the interviews.

In this manner, all of the user's requirements which are explicitly stated in the user's request have been expressed in URL MEMOs for the documentation generator and the manual analysis and, only when possible, in the form of standard URL relations suitable for computer assisted analysis. Notice, that as it used to be, several user requirements have been stated only by implication by such general terms like "present-content", "in compliance with present federal, state, company and union contractual regulations" and so like. The analyst must find out what those statements exactly mean by studying the relevent sources and interviewing competent persons.

3.4 User Requirements Analysis

Before the analyst can proceed to the logical design of the system he should convert all of the user's implied requirements into explicitly stated URL. This step will of course, enhance the statements which already are stored in the data base.

3.4.1 Task Assignment by the Project Leader

It is up to the project leader to review the project task and to select those names which denote the user's implied requirements as well as all kinds of comment entries which should be enhanced and expressed more specifically. If the task is complex enough to be performed by a project team, the project leader should assign to each team member that part of the task he is expected to perform. Assuming the project leader is generally quite experienced in the company's organizational matters (or in case he is an outsider, he should seek advice from the person assigned as liaison between the object organization and the exterior project team) he can also suggest the most appropriate SOURCES in the form of names of existing documents or interviewed people, and also indicate the persons he considers to be authorized to approve the appropriate outcomes of that additional study of the target system's requirements. This can be easily achieved by entering into the URA data base the following statements.

```
DEFINE implied-requirements AS KEYWORD; DESCRIPTION;

To be converted into explicit requirements after studying and
interviewing SOURCES attached to the relevent object names.
Some names of the suggested SOURCES begin with the term
'suggested'.  Other SOURCES can be interviewed after the analyst
has obtained written clearence from the project leader.  The
object names assigned for further investigation can be found by
each analyst assigned to them by the NAME-GENERATION command
with parameter S='TOTAL AND KEY=the-analyst-the-term-assigned'.

The outcome of the study should be approved by managers
indicated by the ATTRIBUTE to-be-approved-by and the ATTRIBUTE
VALUE=management-position, which have been attached to the
object names in question;

APPLIES TO: t-present-pay-check-content,
t-present-payroll-report-cont, t-present-exception-content,
t-present-content-on-term-empl, t-payroll-master-information,
task-payroll-processing, t-companys-computer-facility,
t-if-requested;

DEFINE paycheck-form AS SOURCE, assigned-analyst-name-1 AS
KEYWORD; APPLIES TO t-present-pay-check-content;

GROUP t-present-paycheck-content; ATTRIBUTE IS:
to-be-approved-by payroll-depart-manager-attv; /* In URL
ATTRIBUTES and ATTRIBUTE-VALUES can be defined with the DEFINE
statement but then they cannot be simultaneously related to
```

```
other objects by the APPLIES statement                    */

DEFINE payroll-report-content AS SOURCE,
suggested-payroll-report-clerk AS SOURCE,
assigned-analyst-name-2 AS KEYWORD; APPLIES TO:
t-present-payroll-report-cont;

GROUP t-present-payroll-report-cont; ATTRIBUTE IS:
to-be-approved-by pay-roll-depart-manager-attv;

/* and so on for all concerned names marked with the KEYWORD
implied requirements                                      */
```

This way the project leader has marked all requirement objects which according to his judgement should be investigated more deeply. This KEYWORD name should be maintained until all implied requirements have been analyzed and converted into explicitly stated (and approved where necessary) user's requirements, and then deleted by the URA command DELETE. When the project leader decides that all necessary information has been gathered and approved for a specific object name, the KEYWORD should be dropped by using the DELETE-PSL command for the URL statement with the following form:

```
OBJECT-TYPE object-name-concerned; KEYWORD implied-requirements;
```

3.4.2 Preparation of Interviews by the System Analyst

Now, each assigned analyst can select the system objects assigned to him for further study and generate their hitherto complete description by giving to URA the following commands:

```
NG S='KEY=assigned-analyst-last-name'
/* in order to obtain all possible names, undefined included */
PPS AS /* the option ALL-STATEMENTS (AS) causes also printing of
all permitted statements about the given type of object, which
have not yet been stated*/
```

Before he goes to interview any person which holds the management-positions indicated for some object names, he should study the available organizational documentation. On that basis, he can write down additional URL statements about the assigned objects, which he considers to be the explicit description of the implied user's requirements. The interviews with the assigned persons should not be started before the analyst has exhausted all available sources in the form of available documentation and has prepared a preliminary image how he views the problem. This way, the analyst can later obtain more easily the competent comments and suggestions from the interviewed persons. If the analyst introduced names of new objects and separately described them more in detail (for instance, data GROUPS and ELEMENTS in payroll-report), they also should be marked by the name of the appropriate person to be interviewed and to be asked for approval. They also should be clearly marked and presented as the first draft and subject to changes. That subsequently enables the analyst to separately

generate any kind of URA reports containing the statements concerning only that person. Here is a sample of these kinds of statements, which the analyst made after studying the appropriate forms and associated organizational documentation:

GROUP t-present-pay-check-content; SYNONYM pay-check; CONSISTS OF: employee-name, net-pay, check-number, pay-date; ATTRIBUTE: accompanied-by pay-statement-attv; SOURCE paycheck-form, union-regulations; PROBLEM-DEFINER analyst-last-name; KEYWORD IS: first-draft-subject-to-changes;

GROUP pay-statement; CONTAINED IN: task-payroll-outputs; CONSISTS OF: employee-name, social-security-number, pay-date, check-number, total-hours, gross-pay, total-deductions, net-pay, federal-tax, state-tax, fica-tax; ATTRIBUTE: accompanies pay-check-attv; SOURCE pay-statement-form, union-regulations; KEYWORD implied-requirements; ATTRIBUTE to-be-approved-by payroll-depart-manager-attv; PROBLEM-DEFINER: analyst-last-name; KEYWORD IS: first-draft-subject-to-changes;

Similar statements could be made for other names marked by the KEYWORD implied-requirements. Note that the analyst, while studying the check-form and related regulations, found out that it must be accompanied by pay-statement. Therefore, he felt obliged to state it explicitly, describe the pay-statement and provide it with similar KEYWORD, ATTRIBUTE and ATTRIBUTE-VALUE.

Now, using the URA commands:

```
NG S='ATTR=to-be-approved-by,payroll-depart-manager-attv'
FPS AS
```

He can obtain the complete description of user requirements for all system objects which have to be presented for approval to the payroll-department-manager. This way the interviewed person is bothered only once and furnished only with reports pertaining to their field of responsibility.

3.4.3 Generating URA Reports for the First Interview

(for analysts)

A comprehensive introduction during the interview might be composed of

a) an overall PICTURE report illustrating the most general statements about the system's INPUTS, PROCESS and OUTPUTS, (if applicable) at the highest level of abstraction, followed by

b) main-objective MEMO,

c) FORMATTED-PROBLEM-STATEMENT reports, which represent the verbal comments to the PICTURE reports.

d) more specific breakdown of most important systems objects

(from the point of view of the external user) such as OUTPUTS, INPUTS and sometimes INTERFACES and PROCESSES involved, if the PICTURE reports do not present it clear enough.

The general introductive report usually can be presented to each interviewee, if it does not contradict some security precautions. They should be followed by reports which are the main subject of the interview. The latter should contain only those system's objects of interest to the interviewed person. These groups of URL statements should be printed as separate sets of URA reports concerning each interviewed person. For economic reasons, it is wise (one day in advance) to order the URA to produce them in batch mode during the period of lowest computer charges (usually over night or weekend). The overnight delay in most organizations is insignificant to the analyst. Those objectives can be achieved by the following set of URA commands embedded into the computer installation dependant job order commands not shown in this example:

```
SET DB=ura-data-base-name PNAM=PAYROLL-SYSTEM ECHO=OFF -
     PARAMETERS=OFF PROMPT=OFF
PICTURE N=task-payroll-processing
FPS N=task-payroll-processing AS
PICTURE N=task-payroll-outputs
FPS N=task-payroll-outputs AS
/* The end of general reports */

NG S='ATTR=to-be-approved-by,-
     payroll-depart-manager-attv'
FPS AS
/* and so on, for other management positions indicated by ATTV*/
STOP
```

3.5 Introducing the URL to the Users

(for analysts)

Going to the user for the first interview with carefully prepared URA reports concerning a subject which is easily comprehensible and well-known to the interested user is the best introduction of URL itself to the prospective user. The analyst's introductive oral comments and explanations given for the usually self-explanatory URA PICTURE STRUCTURE and FPS reports simultaneously become an introduction and explanation to URL conventions. The user may not even notice that he is being introduced to a new language; maybe it is only the analysts own rigid and concise way of presenting information processing problems.

The more that the analyst keeps using meaningful, self-explanatory and familiar names to the user, the more he succeeds in the smooth introduction of the prospective user to the URL and the URA reports.

Invariably, all interviews result in some extentions and

alterations of the presented material. The analyst should never forget to ask whether the interviewee has any additional ideas, suggestions, or recommendations that have possibly been missing and could improve the present service rendered by the system and whether he wants credit for them. Sometimes this is possible without increasing the operational cost. Such an approach can gain better cooperation from the user's staff.

In case the target information processing system is intended to support a decision making process, the analyst should obtain from the interviewed decision maker an estimate of expected incremental gains in his objectives which could be attributable to the system's main outputs. At the present level of URL, that could be recorded by means of ATTRIBUTES and ATTRIBUTE-VALUES.

It is important that all changes upon which an agreement has been reached during the interview session are immediately translated and written down by the analyst into the form of URL statements. This lessens the probability of a misinterpretation of the decision maker by the analyst, and minimizes the problems of this nature after the revised document is prepared. If a formal record is taken from each interview, its substance should be worded as much as possible by using the user-defined names of objects and URL reserved (basic and auxiliary) words. Such a record, after minor editorial changes (putting the auxiliary text into /*...*/), when typed on a computer terminal, can simultaneously produce an input file for URA INPUT-PSL command. An overnight production of the appropriate part of the URA reports revised by the analyst, and their presentation to the user the next day after the interview has many significant advantages:

- The user appreciates prompt reaction to his comments by the analyst and that he is asked for verification, corrections and additions.
- He remembers exactly what he agreed upon.
- He gets involved in the development of the system when he feels his comments are seriously included into the design requirements.
- Without a conscious effort, the user starts to think in URL conventions and benefits from them.

After all necessary user requirements concerning the target system in development have been gathered, approved, and stored in the URA data base, the project team using URL/URA can proceed to the second stage of system development - a computer-assisted investigation of the object system and present information processing system.

4. INVESTIGATION AND ANALYSIS OF THE OBJECT SYSTEM AND THE PRESENT INFORMATION PROCESSING SYSTEM

(for analysts)

The target information processing system is intended to serve the object system which may be any kind of organization. Before the systems analyst can proceed to the logical design of the target system he must get acquainted with the object system. The URA Language is particularly well suited to computer assisted description, investigation and analysis of object systems. This includes present information processing systems that are described by any of the known analysis and description methods, such as IBM's Study Organization Plan (SOP) and Business System Planning (BSP), INFOTECH's Structured Analysis and Design Technique (SADT), WARNIER's Logic of Informatics (LI), or the Swedish ISAC's Information Analysis (IA) [see Teichroew, Gackowski, 1977].

According to some authors, system investigation[1] and analysis[2] is a precise diagnosis of the problem or need, and a determination of objectives and performance requirements. This is the case, when the user is unable to state his own needs and objectives. However, even in our example, where the major performance requirement is stated explicitly, there are still several other aspects of the target system which must be determined by investigating the object system. For instance, our hypothetical user left the payroll-inputs as subject to change by the system designer. This is not the case with payroll-outputs where only their form may be subject to changes. However, it makes a considerable difference in respect to the system's performance and scope as to how many types and instances of input data per employee must be processed in order to produce the required paychecks. This can be discovered only by investigating the present information processing system. Similarly, the kind of organizational changes that may and should be made after the implementation of the target system in order to achieve the desired reduction of operation costs, can be determined only by studying the object system within which the target system will operate. Until the necessary facts are gathered, a logical design, a meaningful feasibility study, and an impact analysis of the target system on the object system cannot be undertaken.

However before starting this task, the systems analyst should get acquainted with some useful conventions in assigning the real-world object types to URL objects and with the naming of these objects.

---

[1] Investigation - is used throughout this guidebook in the meaning of study and fact gathering about the object system and the present information processing system.

[2] Analysis - is used throughout this guidebook in the meaning of decomposition of a larger object into smaller parts and studying their interrelations.

4.1 Assigning URL Object Name Types to System Objects

URL requires that each system object named by the analyst must be assigned a certain URL object type. There are 27 types available of which 18 can be defined by their own URL (section headers) statement of the following form:

URL-reserved-word-for-object-type
analyst-defined-name(s)-of-system-object(s);

The other 9 can be defined by the DEFINE statement. In addition, SYNONYMS can be defined for each object type using the DESIGNATE statement.

The assignment of a URL-object type to a system's object via its name is crucial. Statements that can be made about an object and its relationship to other objects are limited to those available in the object section of the URL manual. In some situations the choice of a type for a particular system object is clear; in other situations there may be several legitimate choices. This section discusses the situation in which there are alternatives.

4.1.1 INTERFACES versus PROCESSES and PROCESSORS

In very general terms, a PROCESS is an activity which is part of the target system. A PROCESS is performed on collections of data values represented by names of ELEMENTS, GROUPS, ENTITIES, SETS and INPUTS which USES them to DERIVE new collections of values. The PROCESS can also UPDATE internal data. The data which is used by a PROCESS can come "from" any other PROCESS, and the internal data which is DERIVED can be USED by any other PROCESS.

In contrast, an INTERFACE is a unit outside the boundary of the target system which can produce data for the target system (GENERATE an INPUT) and/or receive data from the target system (RECEIVE an OUTPUT).

An object therefore, should be assigned an INTERFACE type only if it interacts with the target system, namely, that it will RECEIVE or GENERATE data. Otherwise, the organizational units (objects) which will perform PROCESSES (activities) within the target system should be assigned the type PROCESSOR.

4.1.2 INPUTS, OUTPUTS and ENTITIES

INPUTS, OUTPUTS, and ENTITIES are URL types of objects which "contain" or "carry" collections of data values. Conceptually, that type of name can represent both the "container" or the collection of data values contained in that container. Furthermore, the container can be regarded as physical, that is, a card, a tape, a record on a disc, etc., or it can be regarded as a logical construct which may or may not be physically implemented in that form.

A system object should be designated as an INPUT if what is to be specified is a container with data values coming into the target system from outside, i.e., from an INTERFACE.

Another distinguishing characteristic of INPUTS and OUTPUTS is that they are temporary as far as the target system is concerned when interpreted as "containers" of data values. There may be multiple instances of the particular INPUT coming in, but once it is received by a PROCESS, the particular instance disappears.

For example:

INPUT time-card;

implies that the system will receive objects of type INPUT which are called time-card. The number of individual 'time-cards' which arrive is specified by the HAPPENS statement.

Similarly, an object should be designated an OUTPUT if it is a "container" of data values and if it is specified to leave the target system. Again, there may be multiple instances, each one of which has to be GENERATED and each leaves the system. Once individual instances of the OUTPUT have left the target system, they are not considered part of, or accessible to, the target system.

The reasons for distinguishing INPUTS and OUTPUTS from ENTITIES and GROUPS is that:

(1) eventually the physical medium on which they appear and their representation will have to be specified,
(2) the source and destination can be related to INTERFACES,
(3) time and volume can be specified for INPUTS and OUTPUTS but not for GROUPS.

An ENTITY is a "container" of data values; in this respect it is equivalent to an INPUT or OUTPUT. However, it differs from INPUTS and OUTPUTS in that it is internal to the system and it is permanent. Therefore, an individual instance of an ENTITY must be created, i.e., DERIVED, with the exception of ENTITIES which are conceived to contain ELEMENTS with constant values.

Again, the ENTITY may be a "logical" collection of data values or it may be a "physical" collection. When it is designated as a physical collection, it will probably be implemented as a logical record or physical record which is maintained by the system in some way.

Therefore, an object which is a collection of data values that is internal to the target system and is maintained within the system should be designated an ENTITY rather than as an INPUT or OUTPUT.

4.1.3 ENTITIES versus GROUPS and ELEMENTS

An ENTITY is a logical collection of data GROUPS or ELEMENTS that are specified by CONSISTS statements. A GROUP may also CONSIST of other GROUPS or ELEMENTS. But a range of data values can be specified only for ELEMENTS.

The major distinction between ENTITIES and GROUPS lies in that an ENTITY is a container of the values of the ELEMENTS of which it CONSISTS. A GROUP, on the other hand, is merely a notational convenience for naming a collection of ELEMENTS. Whenever the analyst finds that a set of data ELEMENTS appear in a number of situations together, he can save his writing time and analysis time by defining the collection as a GROUP.

Other differences between ENTITIES and GROUPS are the following:

1) GROUPS can be CONTAINED in ENTITIES, INPUTS, and OUTPUTS, but ENTITIES cannot.

2) ENTITIES (and INPUTS and OUTPUTS) can be CONTAINED in SETS, but GROUPS cannot.

3) ENTITIES can CONSIST of GROUPS but not of other ENTITIES. GROUPS can CONSIST of other GROUPS, but of course not of ENTITIES.

4) GROUPS can be used as SUBSETTING CRITERIA of SETS and to IDENTIFY ENTITIES, but ENTITIES cannot. ENTITIES can be RELATED via RELATION statements and have ASSOCIATED data consisting of GROUPS.

5) As far as PROCESSES are concerned, the same statements that can be made about ENTITIES can also be made about GROUPS, though when the ENTITY is used in a statement, the appropriate statement about the ELEMENTS or GROUPS CONTAINED in the ENTITY must also be made.

6) Both ENTITIES and GROUPS can have SYSTEM PARAMETERS associated with the CONSISTS statement. In addition, the ENTITY can have a CARDINALITY and VOLATILITY statement while a GROUP cannot.

The problem definer should specify an object to be an ENTITY when he wishes to refer to a number of ELEMENTS or GROUPS and a collection of values they can contain as a unit.

4.1.4 Definition of SETS

The highest level URL object-type -- which eventually, at the lowest level of decomposition, contains data values -- is a SET. A SET may be divided into subSETs to any level. The subdivision need not be hierarchical since a SET may be a subSET of any number of other SETS and a SET may be subdivided into subSETS in more than one way.

There are two ways in which a SET may be subdivided. The first way is by indicating an ELEMENT name whose value is assigned to be the SUBSETTING-CRITERION[1] on which a SET is to be partitioned, e.g. the SET "customer-information" may be subdivided by "state":

```
SET customer-information;
   SUBSETTING-CRITERION IS state;
```

In this case "state" is the name of an ELEMENT. If more than one ELEMENT is given in the statement, the specification of whether all, some, or any are meant should be included in the DESCRIPTION.

The second way is to define an object type SUBSETTING-CRITERION. In this case that object is not an ELEMENT and can be defined in more detail in a DEFINE section. This method should be used when the criterion for subdivision is not an ELEMENT, i.e., if a SET is to be subdivided on the basis of one or more ELEMENTS which are not contained in the ENTITIES in the SET and these ELEMENTS should be used as the SUBSETTING-CRITERION.

In top-down development, a SET should be divided into subSETS until it is clear that ENTITIES have been reached. For example, in a personnel system the name assigned to all of the stored information might be "personnel-information". This may be subdivided into "salaried-personnel-information" and "hourly-personnel-information" and into "current-personnel-information" and "inactive-personnel-information" and so on. At some point, a level will be reached at which the units should be defined as ENTITIES.

In bottom-up definition the ELEMENTS would first be defined individually, then named as being contained in particular ENTITIES. These would then be named as being contained in SETS which could be made to be contained in other SETS and so on.

Generally the point at which the subdivision of SETS into subSETS is replaced by the definition of ENTITIES is where the contents of the ENTITY are well enough defined so that one can think of a number of instances of the ENTITY containing the same ELEMENTS. For example, consider the definition of a URL data base which will be used in the design of a physical system. During this process the data base will be updated as new information is obtained. Initially the data base may be named as:

```
SET: physical-design-information;
```

---

[1] For the purposes of this description GROUPS may be regarded merely as a shorthand way of referring to two or more ELEMENTS. Therefore where ever the word ELEMENTS appears in this description, implying two or more may be used, the word GROUP can be substituted.

The information may be then thought of as consisting of information describing the computer hardware that will be used, the data base of the target system, the noncomputerized procedures, the software, etc. Since different data elements will be used to describe each one of these types of information these could be defined as ENTITIES or as SETS. If it is clear that the same data elements will be used for all instances of, for example, the computer hardware description it would be appropriately defined as an ENTITY. On the other hand if at this point, it appears that hardware may be further subdivided into CPUs, terminals, etc. which are likely to be described by different elements it would be better to define the different types of information as subsets as follows:

```
SET physical-design-information;
SUBSETS ARE hardware-information,
data-base-information, noncomputerized-procedures,
software-information;
```

At this point the hardware information may be thought of as consisting of information about CPUs, about auxilary memories, terminals, etc. If each of these now can be described by the same collection of ELEMENTS, these could be defined as follows:

```
SET hardware-information;
CONSISTS of CPU-information,
auxilary-memory-information, terminal-information;
```

With this definition URA would treat these three new names as UNDEFINED since they could be either GROUPS or ENTITIES. If the problem definer is uncertain as to whether he wants to call them SETS or ENTITIES he can leave them in this form until he specifically determines the data content and verifies that it will in fact be homogenious.

In URL a particular object of type SET is defined in detail by specifying the names of only ENTITIES, only INPUTS or only OUTPUTS.

In particular, a SET may consist of ENTITIES. This definition means that the SET in the target system will contain some number of instances of each of the named ENTITIES.

In the target system there may be an arbitrary number of the instances, including zero, of each one of the ENTITIES. The particular collection of data objects which make up an ENTITY may vary over time in the target system but the types of data objects which may occur is fixed by the definition of the SET.

4.1.5 ENTITIES versus SETS

ENTITIES and SETS are equivalent as far as data operations are concerned since both ENTITIES and SETS may be USED, DERIVED or UPDATED by PROCESSES. They are also equivalent in information about their size since CARDINALITY is specified for each by a SYSTEM-PARAMETER and system dynamics information in that VOLATILITY, in each case, is described by a comment-entry.

ENTITIES and SETS however, differ in four very important respects:

1) ENTITIES cannot be subdivided into other ENTITIES, while SETS can be subdivided into subSETS in many ways.

2) ENTITIES may CONSIST of GROUPS or ELEMENTS while SETS cannot contain these data object directly.

3) ENTITIES may be IDENTIFIED by ELEMENTS.

4) ENTITIES may be involved in RELATIONS.

(1) and (2) make it possible to use an ENTITY in the same sense that E.F. Codd uses "relation", i.e., as a table with many rows identified by the IDENTIFIER, and the same set of columns, namely the ELEMENTS of which the ENTITY CONSISTS.

(4) makes it possible to describe requirements of stored data without implying a physical data organization. This is further described in the section on RELATIONS.

SETS and ENTITIES may be USED, DERIVED and UPDATED. Whenever one of these statements is made, it is implied that at least one ELEMENT or GROUP is USED, DERIVED or UPDATED.

An analysis of existing overlaps can be obtained from the URA CONSISTS COMPARISON report.

4.1.6 Defining RELATIONS Between ENTITIES

URL permits the definition of an object-type called RELATION. The basic purpose of this facility is to permit the logical definition of data as viewed by users of the target system without implying a particular physical implementation. The problem definer can concentrate on describing the logical view of the data. The physical designer can be certain that he has received all the logical requirements in a form he can use to evaluate physical implementation alternatives.

The following example will illustrate the use of RELATIONS.

A university wishes to develop a computer aided registration system. The purpose of this sytem is to 1) enroll students in courses of their choice subject to constraints on eligibility, 2) to produce hard-copy lists of the courses each student is enrolled in, and 3) generate a list of all students enrolled in

a course for each instructor. In addition various management reports are to be produced.

The university has approximately 40,000 students enrolled in each term. It offers approximately 10,000 courses. Students are enrolled in an average of five courses and are on the waiting list for one additional course.

During the registration period which starts eight weeks before the beginning of a term and ends four weeks after the beginning of a term, there is a great number of changes. Departments add new courses and cancel others in response to student interests. Students change their course preferences based on availability, time conflicts, etc.

The new system is required to respond to these changes quickly and to follow through the consequences of any one change.

In defining the requirements of this sytem, the need for "stored" data will require the definition of two ENTITIES; one for students and one for courses.

```
ENTITY student-information;
   CONSISTS OF student-identification,
               other-student-information:
IDENTIFIED BY     student-identification;
ENTITY course-information;
   CONSISTS OF course-identification,
               other-course-identification;
   IDENTIFIED BY  course-identification;
```

The other information in each case will consist of the data necessary to describe each student and each course respectively.

Courses and students are related by: enrollment, waiting list, grade, etc.

If the relational information were included in the ENTITY student-information, the system may well be implemented with course-information in the student record. If the system will always be used to produce output containing all or some courses in which a student is enrolled or on the waiting list, these may be ASSOCIATED-DATA.

4.2 Naming Conventions of System Objects

URL requires that all objects have unique names. This is one of the features which makes a description in URL more precise and unambiguous. In the URA analysis, the name itself is not important - it could just as easily be replaced by an arbitrarily assigned number - it is the relationships in which the name appears that is important.

For human understanding, however, the name itself is very important since it permits the reader to associate the object with something in the real world with which he is familiar - the

name means something to him. It is, therefore, very desirable to ensure that the names are as meaningful as possible and that the meaning be as consistent as possible to all readers.

This can best be accomplished by establishing and consistently following conventions and standards. Unfortunately, there are no universally accepted conventions and standards. This section contains some suggestions and guidelines.

Since the different types of objects play different roles, the guidelines for naming for each type are different.

Since each URL object must be of a specific type it is usually desirable to be able to identify the type from the name itself. This is incorporated in the guidelines given for each type of name below.

Names are essentially meaningful codes and therefore, the methods for developing such codes are applicable to the naming problem.

Naming conventions may include the categories of terms which may be used and the order in which they are to be used. In some cases they may also include a list of available terms. The collection of terms should always reflect the terminology that is customary and familiar to the eventual user's staff and management, and not the analysts preferences, for they are first of all communication media with the system user.

4.2.1 Data ELEMENTS, GROUPS, ENTITIES, and SETS names

A data ELEMENT should be named by a noun which in plain English designates the collection of data values it is intended to represent, for instance: name. If there are several collections of values of this type which the analyst wants to distinguish he can modify the noun by any adjective, for instance: manager-name, employee-name, customer-name.

A higher level collection of data ELEMENTS should be named similarly by using more abstract expressions with a broader meaning such as:

-- address-sp -- sp for a data GROUP which CONSISTS OF: recipients-name, house-number, street-name, locality-name, state-name, zip-code,

-- employee-record--for an ENTITY which CONSISTS of data GROUPS: employee-address and other-information,

-- payroll-transactions--for a SET which CONSISTS OF data ENTITIES: time-card, job-card, leave-of-absence-card,..,

-- payroll-sets--for a SET which SUBSETS ARE: payroll-transactions, payroll-master, payroll-reports, and which is in turn a SUBSET OF fiscal-data-base -- a higher level SET.

Generally, names of SETS used to represent a master file or a data base should have the last term like -master and -data-base respectively.

ENTITIES are best thought of as logical records and therefore, can have a last term such as -information or -record if a more physical connotation is intended.

The OUTPUTS and INPUTS should be named similarly as above, but in addition with the ending term such as -report, -output or -input and so like respectively.

Names of INPUTS may also end in a carrier type term, e.g., -card, -message, -document.

4.2.2 Names of RELATIONS between ENTITIES

URL enables the analyst to define some logical relations which exist between different kinds of data ENTITIES. The recommended naming convention is to connect the names of the separate ENTITIES with the preposition "to". For instance, department-to-employee(s), in plural, if a one to many CONNECTIVITY intended.

4.2.3 PROCESS Names

A PROCESS is an object which represents an action, e.g., deriving data from other data. The basic part of the name should be a verb which describes the action. Usually, the object of the action should be included. For example:

calculate-gross-pay

is a meaningful description of a PROCESS giving both the action "calculate" and the object or the result "gross-pay".

When this convention is followed it is usually clear that the object is a PROCESS. If desired, however, the type can be included, e.g.:

calculate-gross-pay-process.

In assigning the action part it is usually desirable to avoid words which have several meanings or which may be both nouns and verbs. For example:

logical-design

is a name which is subject to several interpretations. A better name would be:

design-logical-system

4.2.4 SYSTEM-PARAMETER and ATTRIBUTE-VALUE Names

SYSTEM-PARAMETERS or ATTRIBUTE-VALUES should use self-descriptive names for the values they represent or are related to, for instance: twenty-five or number-of-employees respectively.

4.2.5 KEYWORD and ATTRIBUTE Names

An ATTRIBUTE may have only one ATTRIBUTE VALUE for a given object and, therefore, should be used when the classification is mutually exclusive. If an object may possess several attribute values these must be defined as KEYWORDS.

Both KEYWORDS and ATTRIBUTES are used to describe objects and to facilitate subsequent retrieval of the objects that they modify. The ATTRIBUTE name, therefore, should be a meaningful English name for any attribute that can be of interest to the prospective user of the system. It may be a symbolic name when it is intended only for internal use within the project team, but then the symbols should be subject to some separate standard convention used by the team. This is relevant both to ATTRIBUTES and KEYWORDS because KEYWORD names represent only a simplified notation for ATTRIBUTES which do not have multiple ATTRIBUTE-VALUES but simply exist or do not exist for some system objects.

4.2.6 Other Useful Conventions in Naming System Objects

The project task, the object (organizational) system, the present system, and, eventually, all of the target system are investigated, analyzed, designed and described during the various stages of system development. In URL/URA all the described system objects are stored in the same data base in order to enable a flexible comparison and analysis. Many of the objects are often similar or are nearly the same semantically, with the only difference being that they are part of the description of either the project task, the object system, the present system, or the target system. For retrieval purposes they can easily be recognized by assigned KEYWORD names such as task, object, present, or target. Still, it does not prevent a misuse of the same system object name in all four system types. That, of course, could cause the creation of unwanted relations between objects in which abstraction should belong to only one of the different systems.

This may be easily assured for the whole project team by adopting a standard naming convention for all objects belonging exclusively to one of the mentioned system types in the following manner:

- all names of objects belonging to the project task description should use a prefix, like task- or t-,
- all names of objects belonging to the object system

description should use a prefix, like object- or ob-,

- all names of objects belonging to the present system description should use a prefix, like present-, pr-, or is-,

- all names of objects belonging to the target system description must not use any prefix. The names chosen should be the simplest ones without any artificial terms in order to assure the highest degree of comprehensiveness to the target user when they are presented in the final reports.

Some naming conflicts can occur within the description of the same system. URL, for the sake of unambiguity, does not allow the use of the same name for different types of objects though it is entirely legible and legal in natural languages and in real life situations. For instance, semantically, the same department-manager can be in URL

- an INTERFACE which RECEIVES an OUTPUT,
- a PROCESSOR which PERFORMS a decision-making PROCESS,
- an ATTRIBUTE-VALUE or a KEYWORD or a SOURCE name for instance in a URL statement.

```
ATTRIBUTE IS to-be-approved-by department-manager;
KEYWORD IS department-manager;
SOURCE IS department-manager;
```

URA would not accept such a use of the same name in any pair of the above mentioned situations. This naming conflict can again be solved by adopting a naming convention. In cases where it is necessary to use the same analyst defined name for different URL types of system objects, the names should be modified by adding to the chosen name a suffix made of one of the permitted abbreviations of the appropriate reserved word, for instance, department-manager-intf, department-manager-prcr, department-manager-attv, department-manager-key, department-manager-src.

In general, it is wise to avoid those names and abbreviations of names which are not meaningful to the system user. Therefore, the system department's standards and conventions should strictly prohibit the analyst to introducing any artificial terms (prefixes or suffixes) that are not part of user-friendly names, except in the case of naming conflicts.

These conventions should be defined before the description of a system in URL is begun. The project leader will usually establish these conventions.

KEYWORDS and ATTRIBUTES may be used to define the characteristics of objects which will have meaning in the URA report generation process. For example, KEYWORDS level-1, level-2, etc. could be assigned to objects defined at the various levels of the requirements description. These KEYWORDS could then be used to retrieve information pertaining to a

single level.

Other conventions may deal with the use of names and SYNONYMS. For example, if organizational-units or documents within an organization have both names and codes, the use of them as INTERFACES or PROCESSORS or INPUTS may require that the codes be used as SYNONYMS so that information may be retrieved by code.

Standards may also dictate that a RESPONSIBLE-PROBLEM-DEFINER be specified for each URL object defined, or that security informaiton (either for target system objects or for URL sections) be defined in a particular way.

The following references are relevant:

Fraser, 1971; IBM Data Dictionary; Butterworth, 1975; Simmons, 1976; Florentin, 1976; McNamee, 1973.

4.3 Task Assignment for the Project Team

(for Project Leaders)

Following again our previous payroll example, the project leader should define, mark, and assign to the analysts all objects of the target system and the present-payroll-system which should be investigated. In the example from section 3, the most important and interesting objects seem to be 1) the present-payroll-inputs, 2) the present organizational-units (INTERFACES) which GENERATE parts of the present-payroll-inputs (perhaps some of them could deliver them in computer readable form), 3) the present PROCESSORS, PROCESSES and RESOURCES involved with present-payroll-processing. If the project staff is not familiar with the object system and payroll system at all, then the project leader should decide to investigate both the object system and the present-payroll-system.

There are also some general but useful rules (see Figure 4.1 by Burch and Hod, 1975):

1) The less adequate the present information processing system, the less time the analyst should devote to investigating it, but the more the object system is of interest.

2) The less modification and the smaller the improvement scope, the more time the analyst should devote to investigate the present information processing system and the less to the object system.

Figure 4.2 presents a generalized structure chart for use of URL/URA to describe systems.

Such an investigation and analysis assignment can be prepared by the project leader in the form of following URL statements

concerning our example:

```
DEFINE to-investigate AS KEYWORD; DESCRIPTION;
To investigate and describe the objects denoted by names
marked with this KEYWORD, by studying and interviewing
SOURCES attached to them;/* rest like for the KEYWORD
implied-requirements in s.3.4.1 */
APPLIES TO: present-payroll-inputs,
present-prs-inputs-interfaces, present-payroll-processors,
present-payroll-activities, present-payroll-resources;

INPUTS present-payroll-inputs; ATTRIBUTES ARE:
to-be-described-till data-elements, where received-from;
SOURCE ARE present-forms, payroll-input-checker; KEYWORD IS
assigned-analyst-name-1;

INTERFACES present-prs-inputs-interfaces; ATTRIBUTES ARE;
to-be-also-described as-processors-if-necessary,
to-describe-the-feasibility-of computer-readable-inputs;
SOURCES ARE orgu-managers; KEYWORD IS
assigned-analyst-name-2;

PROCESSOR present-payroll-processors; ATTRIBUTES ARE:
to-be-described-with all-performed-activities,
and-with-resource-usage-by SOP-method; SOURCES ARE
all-involved-employees; KEYWORD IS assigned-analyst-name-3;
PERFORMS present-payroll-activities; CONSUMES
present-payroll-resources AT RATE OF x-units PER
activity-unit;
/* Note, that the definition of object types, investigation
ATTRIBUTES, KEYWORDS and SOURCES for
present-payroll-activities and present-payroll-resources is
not necessary, because they are closely related to the
above defined PROCESSOR and defined by the above stated URL
relations */
```

In this case, the procedure of using URL/URA for the investigation of the object system and the present-payroll-processing-system is similar to that described in section 3.

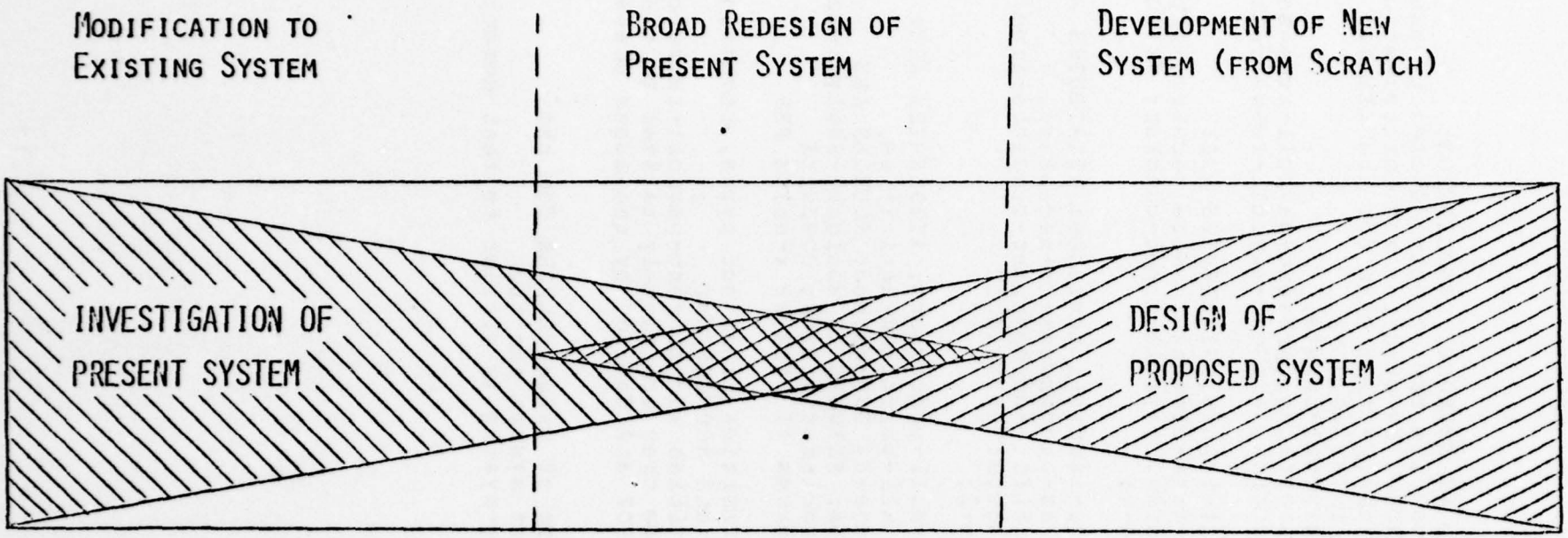
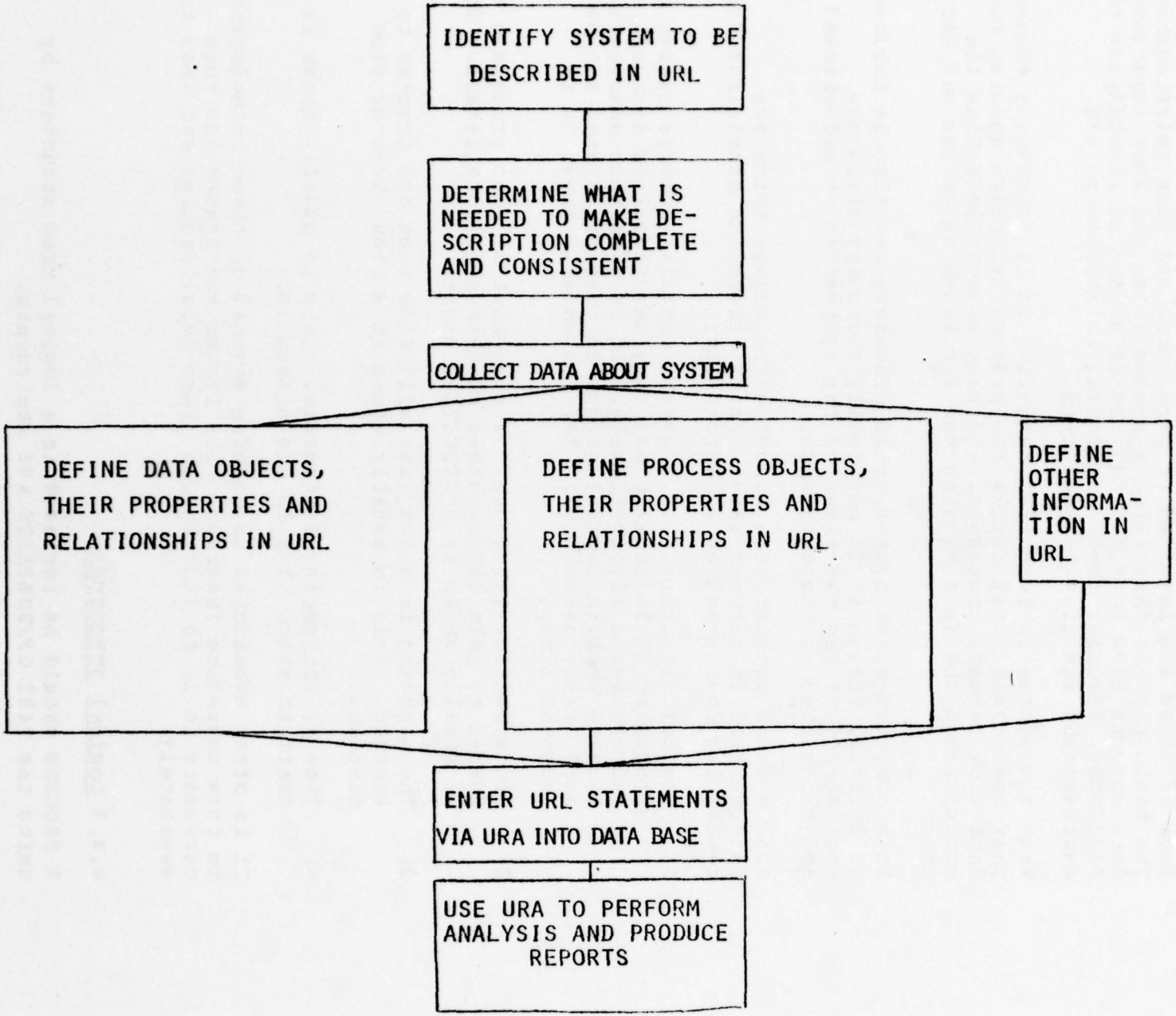


FIGURE 4.1 DEGREE TO WHICH THE SYSTEMS ANALYST INVESTIGATES THE PRESENT SYSTEM. (AS ONE MOVES FROM LEFT TO RIGHT, THE NEED TO INVESTIGATE THE PRESENT SYSTEM LESSENS UNTIL IT BECOMES INAPPLICABLE, WHERE A NEW SYSTEM IS DEVELOPED FROM SCRATCH.) CONVERSELY, NOTE THE DOVETAILING PROCESS OF PROPOSED SYSTEM DESIGN (CROSS-HATCHED PORTION). (BURCH, J. G., AND HOD, N., INFORMATION SYSTEMS: A CASE-WORKBOOK, JOHN WILEY & SONS, INC., 1975, P. 37.)

FIGURE 4.2 STRUCTURE CHART FOR USE OF URL/URA TO DESCRIBE SYSTEMS



4.4 Defining the PROCESS Structure

The major types of objects in systems are processes and data. Each of these are involved in structural and flow relationships. The basic task in describing a system after the most important data objects have been defined can be stated as identifying the PROCESSES, defining their structural relationships and representing the flow between them.

When the system is described manually, it is frequently argued that one should first define the process structure showing the data flow between processes, and then separately define the structure of the data showing the processes which use and derive it.

In a computer-aided system, this procedure can also be followed if desired, though it is not really necessary since the consistency of the two views of the system can be established and maintained in the data base.

Most systems consist of a number of processes which are interrelated in various ways. In URL there are provisions for describing four types of relationships:

1) Logical Structure. If a system contains a large number of processes it is usually not very meaningful to list all of the processes alphabetically. It is much more meaningful to have them in logical groups so that the system can be more easily understood. This is usually shown in an indented list.

2) Decomposition Structure. A statement of what processes are needed to help define other processes in more detail. This is usually shown in a STRUCTURE chart.

3) The sequence in which data will flow from one process to another. This is usually shown in a flow chart or flow diagram.

4) Time or Triggering Statements. This is usually shown in a chart in which time is one dimension.

It is often beneficial to combine several of these relationships to form composite diagrams. The larger the system the more necessary it is to distinguish these relationships and show them separately.

4.4.1 Logical STRUCTURE

A PROCESS should be included in a logical tree structure by using the PART OF/SUBPARTS ARE statements.

The purpose of these statements is to allow a logical table of contents to be produced in which each PROCESS appears once, and only once, and which allows the system to be studied in general or in detail, or to quickly identify any particular logical

area.

4.4.2 PROCESS Decomposition

Each PROCESS definition contains a description, in terms of its decomposition into lower level PROCESSES. This is accomplished using the following statements: DESCRIPTION, UTILIZIES/UTILIZED, USES/DERIVES/UPDATES, PROCEDURE.

DESCRIPTION

The DESCRIPTION should contain information which can be verified by the target-system user. The systems analyst and the user can check that the description is consistent with other URL statements. The following, in text form, should be included in the description:

a) A brief statement of the purpose of the PROCESS. This should usually be one simple sentence that can be consistent with, and elaborate on, the name of the PROCESS.

b) A statement of the logical grouping in which the PROCESS appears based on the PART OF/SUBPARTS ARE statement.

c) A statement of the decomposition in which the PROCESS appears based on the UTILIZED/UTILIZES statement.

UTILIZES/UTILIZED and TRIGGERED

Each PROCESS should be UTILIZED or TRIGGERED by at least one other PROCESS or TRIGGERED by some EVENT. These relationships should be included in the PROCESS definition.

Each PROCESS definition contains a list of the PROCESSES which it UTILIZES; these are given in the UTILIZES statements. These statements should be consistent with the order of statements in the PROCEDURE statement.

If a system has only a tree invocation structure, the UTILIZES statement should be used. The SUBPARTS statement need not be used since a STRUCTURE report can be obtained by the UTILIZATION ANALYSIS report.

PROCEDURE

The PROCEDURE comment entry should be used to show the steps followed in executing the PROCESS. Usually the steps should correspond to the PROCESS which is UTILIZED.

This statement is mandatory for the lowest levels of PROCESSES as a specification for further programming efforts.

4.5 Using URL/URA for Various Documentation Methods

(for analysts)

Examples of the use of URA to produce doecumentation according to various standards is given in Part V of the URA Users Manual.

4.6 Checks Carried Out by URA and the Analyst

The completed investigation and analysis of the object system and the present information-processing systems must be checked for preciseness, consistency and completeness. The logical design, the physical design and the intermediate and final documentation of the system must also be checked similarly. Therefore, the checks performed and facilitated by URL/URA and described in this section pertain to all stages and steps of system development and documentation.

Preciseness, consistency, and correctness are all checked by the UR Analyzer as new information is added to the data base or data is modified in it. URA can produce several hundred diagnostic and error messages. Each is identified by a number. The complete list is given in the "H6180/Multics/URA User's Manual" in numerical order to facilitate correction. In the discussion below, the error messages are analyzed in terms of how they contribute to good investigation, analysis, design and documentation.

4.6.1 Checks Related to Preciseness Carried Out by URA

A considerable portion of the error detection facilities in the Analyzer are used to check the "preciseness" of new URL statements being added to the data base. (This is done during the execution of the URA command INPUT-PSL.) The Analyzer checks that the syntax is correct and that the user-defined names given in the new statements are consistent with names already in the data base. If either of these conditions fail, an error diagnostic is generated by the Analyzer to inform the user that the information to be stored in the data base was ambiguous or inconsistent with the information already there. No ambiguous or inconsistent information is permitted into the data base.

a) Syntax Errors

Violating any of the syntax rules of URL will cause the Analyzer to generate one or more error diagnostics. Typical syntax errors are:

- use of illegal characters,
- misspelling of URA reserved words,
- omission of a semi-colon to terminate a line.

Table 4.1 presents a complete list of diagnostics produced when a syntax error is encountered.

Table 4.1 URL Syntax Error Messages

Error Number		Diagnostic Message
2	NLEX :	NAME TOO LONG
3	NLEX :	'EOF' NOT FOUND BEFORE END-OF-FILE
4	INDBS :	ERROR OPENING DATA BASE FOR - data-base-name
5	NLEX :	END-OF-FILE IN MIDDLE OF COMMENT
7	SCAN :	ILLEGAL CHARACTER - IGNORED
10	REDUCE :	NO APPLICABLE PRODUCTION - SYNTAX ERROR - START SKIPPING
11	STACK :	ILLEGAL SYMBOL PAIR - SYNTAX ERROR - START SKIPPING
16	COMENT :	END-OF-FILE IN COMMENT ENTRY
17	SKIP:	END OF FILE WHILE SKIPPING
90	RWLIST :	SSCN IS ONLY LEGAL TYPE IN DEFINE SECTION WHICH CAN BE MAINTAINED
114	VLIST :	ONLY SINGLE VALUE OR RANGE ALLOWED - IGNORED
116	OTHERS :	VALUES ONLY LEGAL FOR ELEMENT, SYSPAR, OR ATTRIBUTE-VALUE
201	PLIST :	NAME NOT PART OF HEADER
225	RWLIST :	CANNOT HAVE KEYWORD FOR KEYWORD
228	RWLIST :	CANNOT HAVE SECURITY FOR SECURITY
229	RWLIST :	CANNOT HAVE SOURCE FOR SOURCE
231	RWLIST :	SYNONYMS ONLY APPLIED TO FIRST NAME
232	APPLES :	APPLIES STATEMENT ILLEGAL WITH THIS NAME TYPE
266	ILLST	ILLEGAL STATEMENT IN THIS SECTION
512	RWLIST:	CANNOT HAVE TRACE-KEY FOR TRACE-KEY

b) Naming Errors

It is very important that once a name is defined and has an associated name type along with it (e.g. PROCESS or SET), the name can only be used in the context in which it was defined. Therefore, a name defined to be a PROCESS cannot be also used to define a GROUP of data. Likewise, only those relationships specified by the "User Requirements Language, Language Reference Manual" can be used to relate two objects. For example, a USES relationship between two PROCESS names is not allowed and any attempt to specify this would cause the Analyzer to generate the diagnostic: MUST BE ELEMENT, GROUP, INPUT, ENTITY, OR SET. Table 4.2 presents a complete list of the errors that can be encountered in relation to names.

4.6.2 Checks Associated with Consistency Carried Out by URA

As URL statements are added to the data base, the Analyzer also checks that the new relationships specified are consistent with the information already in the data base. In the previous section, the Analyzer was shown to check that once a name was defined to be a given name type, it could not be used in a conflicting context (i.e., as a different name type). The

Analyzer also checks that the relationships specified for a given name do not conflict. For example, if an ENTITY was defined to have a CARDINALITY of 100, it would be illogical to also say that its CARDINALITY is 50. The Analyzer detects these kinds of inconsistencies. The Consistency Error Messages are listed in Table 4.3. Table 4.4 presents the various inconsistencies detected by the Analyzer according to name type and relationships within the existing URL system description.

Table 4.2 URL Name Error Messages

Error Number		Diagnostic Message
25	HEAD	INVALID HEADER STATEMENT - STATEMENTS WILL BE IGNORED
51	RWLIST	MUST BE SUBSETTING-CRITERION NAME
53	OPTRW:	NAME LIST TOO LONG - REST IGNORED
100	NLIST2	TOO MANY ATTRIBUTE VALUE PAIRS IN SINGLE STATEMENT
101	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT
102	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT
120	SNAMET	NAME ALREADY USED IN DIFFERENT CONTEXT
202	NLIST	NAME PREVIOUSLY USED DIFFERENTLY - IGNORED
204	DEFN	NAME ALREADY USED IN DIFFERENT CONTEXT
206	SETSYN	UNABLE TO MAKE SYNONYM - TOO COMPLICATED
207	SETSYN	CANNOT BE MADE SYNONYM - DIFFERENT TYPES
209	CHKCON	STACK OVERFLOW WHILE WALKING CONSISTS STRUCTURE
210	PRTNUN	NO NAMES IN DATA BASE
211	OTHERS	NAME MUST BE ENTITY NAME
216	OTHERS	NAME MUST BE ENTITY NAME BEFORE VIA
217	OTHERS	NAME MUST BE RELATION AFTER VIA
220	SETSYN	CANNOT MAKE A NAME FOR ITSELF
221	NLIST	TOO MANY NAMES - REST IGNORED
222	CONSUM	NAME MUST BE AN INTERVAL
224	OPTRW	NAME MUST BE AN ELEMENT OR CONDITION
233	DEFN	TOO MANY NAMES IN DEFINE HEADER - REST IGNORED
234	OPTRW	NAME ALREADY USED IN DIFFERENT CONTEXT
235	OPTION	NAME ALREADY USED IN DIFFERENT CONTEXT
236	OPTION	NAME LIST TOO LONG - REST IGNORED
240	APPLES	KEYWORD CANNOT APPLY TO KEYWORD
241	APPLES	MAILBOX CAN ONLY APPLY TO PD
246	APPLES	SECURITY CANNOT APPLY TO SECURITY
247	APPLES	SOURCE CANNOT APPLY TO SOURCE
248	APPLES	MEMO CANNOT APPLY TO MEMO
257	FORMSL	NAME NOT IN DATA BASE -
267	ILLST	NO CURRENT SECTION
269	NLIST	NAME LIST TOO LONG, REST IGNORED
511	APPLES	TRACE KEY CNANOT APPLY TO TRACE KEY

Table 4.3 URL Consistency Error Messages

Error Number		Diagnostic
22	RWLIS2	SAME ATTRIBUTE ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE
43	OTHERS	CARDINALITY ALREADY GIVEN AS SYSPAR
44	OTHERS	CARDINALITY ALREADY GIVEN AS DIFFERENT VALUE
60	APPLES	SECOND MAILBOX FOR PD ILLEGAL
61	RWLIST	ALREADY PART OF SOMETHING ELSE
62	RWLIST	SECOND PD FOR THIS NAME ILLEGAL
63	RWLIST	ALREADY PART OF SOMETHING ELSE
115	VLIST	MIN NOT LESS THAN MAX - IGNORED
117	OTHERS	DIFFERENT VALUES ALREADY GIVEN
205	SETSYN	ALREADY SYNONYM FOR SOMETHING ELSE
212	OTHERS	RELATION ALREADY EXISTS BETWEEN TWO OTHER ENTITIES
213	OTHERS	CAN HAVE ONLY ONE CARDINALITY
214	OTHERS	CONNECTIVITY ALREADY GIVEN FOR THIS RELATION
215	RWLIS2	ALREADY CONTAINS WITH DIFFERENT SYSTEM PARAMETER
218	OTHERS	RELATION ALREADY EXISTS BETWEEN DIFFERENT ENTITY
223	RWLIS2	INCONSISTENCY IN "HAPPENS WITHIN" STATEMENT 91
263	RWLIS2	INCONSISTENCY IN "HAPPENS AFTER" STATEMENT
264	RWLIS2	INCONSISTENCY IN "HAPPENS WITHIN" STATEMENT
265	RWLIST	CONNECTION ALREADY EXIST WITH DIFFERENT VALUE OR NAME
514	RWLIS2	ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE
560	RWLIS2	INCONSISTENCY IN CLASSIFICATION STATEMENT
561	RWLIS2	INCONSISTENCY INCECURITY-ACCESS-RIGHT STATEMENT
562	RWLIS2	INCONSISTENCY IN RESOURCE-USAGE STATEMENT
563	RWLIS2	INCONSISTENCY IN RESOURCE-USAGE-PARAMETER-VALUE STATEMENT
569	RWLIS2	INCONSISTENCY IN CONSUMES/CONSUMED STATEMENT

CONSISTENCY ERRORS

	System Flow	System Structure	Data Structure	Data Derivation
RWE		61,63		
INPUT		61,63		
OUTPUT		61,63		
ENTITY			212,218	
RELATION			212,218	
PROCESS		61,63		
OTHER	205,206	205,206	205,206 560,561	90,205,206

TABLE 4.4

	System Size	System Dynamics	System Properties	Project Management
RWE				62
INPUT	215			62
OUTPUT	215			62
SET	43,44 213,215			62
ENTITY	43,44 213,214			62
RELATION	43,44 213,214			62
GROUP	215			62
ELEMENT	117,115			62
PROCESS				62
INTERVAL	215	223,263,264		62
SYSTEM PARAMETER	265,115	223,263,264		62
EVENT		223,263,264		62
CONDITION				62
CONDITION				62
OTHER	205,206	205,206	22,205,206	60,62,205,206

TABLE 4.4 (continued)

4.6.3 Consistency and Completeness Checks Based on URA Reports and Carried Out by the Analyst

At some point in time, the Analyst may want to check the state of consistency and/or completeness in the development of the problem statement. The Analyst can perform these checks by inspection of various reports available from the Analyzer. This technique is possible because all information specified in the data base can be presented via one or more reports. Even though the Analyzer has checked all inputs to the data base for syntax and consistency errors, the problem statement is not always in a correct state. It is the role of the Analyst to determine whether it is totally "consistent" or "complete".

Table 4.5 presents a summary of all consistency and completeness checks to be carried out by the Analyst.

Table 4.6 presents a summary of the benefits of particular URA reports in identifying inconsistencies and incompleteness in the problem statement.

Table 4.5 Summary of Completeness Checks to be made by Analyst

I) SYSTEM FLOW
   a) All INTERFACES should GENERATE some INPUT, RECEIVE some OUTPUT, or be RESPONSIBLE for some SET.
   b) All INPUTS should be GENERATED by at least one INTERFACE.
   c) All INPUTS should be RECEIVED by at least one PROCESS.
   d) All OUTPUTS should be GENERATED by at least one PROCESS.
   e) All OUTPUTS should be RECEIVED by at least one INTERFACE.

.....

II) SYSTEM STRUCTURE
   a) All PROCESSES without SUBPARTS should have PROCEDURES.
   b) SETS with SUBSETS should have SUBSETTING-CRITERIA.
   c) All INPUTS without SUBPARTS should be broken down via the CONSISTS statement.
   d) All OUTPUTS without SUBPARTS should be broken down via the CONSISTS statement.

.....

III) DATA STRUCTURE
   a) All ELEMENTS should be available from an INPUT or from a ENTITY, or DERIVED by some PROCESS.
   b) All SETS should CONSIST of INPUTS, OUTPUTS or ENTITIES.
   c) All ENTITIES should be broken down via the CONSISTS statement.
   d) All INPUTS should be broken down via the CONSISTS statement if there are no SUBPARTS.
   e) All OUTPUTS should be broken down via the CONSISTS statement if there are no SUBPARTS.
   f) All RELATIONS should have a BETWEEN statement.
   g) All GROUPS should be composed of ELEMENTS.

.....

IV) DATA DERIVATION
- a) All ELEMENTS should be USED, UPDATED and/or DERIVED by at least one PROCESS.
- b) All PROCESSES should acquire information by RECEIVING, USING or UPDATING.
- c) All PROCESSES should produce information by GENERATING, DERIVING, or UPDATING.
- d) All SETS should be USED, UPDATED or DERIVED by at least one PROCESS.
- e) All SETS should have a DERIVATION statement.
- f) All ENTITIES should be USED, UPDATED or DERIVED.
- g) All ELEMENTS within an INPUT should be USED.
- h) All ELEMENTS within an OUTPUT should be DERIVED.
- i) All ELEMENTS within an ENTITY should be USED, UPDATED or DERIVED.
- j) All RELATIONS should be MAINTAINED by at least one PROCESS.
- k) All RELATIONS should have a DERIVATION statement.

.....

V) SYSTEM SIZE AND VOLUME
- a) All EVENTS should have a HAPPENS statement.
- b) All PROCESSES should have a HAPPENS statement.
- c) All SETS should have a CARDINALITY statement.
- d) All SETS should have a VOLATILITY-SET statement.
- e) All SETS should have a VOLATILITY-MEMBER statement.
- f) All ENTITIES should have a CARDINALITY statement.
- g) All ENTITIES should have a HAPPENS statement.
- h) All INPUTS should have a HAPPENS statement.
- i) All OUTPUTS should have a HAPPENS statement.
- j) All RELATIONS should have a CARDINALITY statement.
- k) All RELATIONS should have a CONNECTIVITY statement.

.....

VI) SYSTEM DYNAMICS
- a) Each EVENT should be associated with at least one CONDITION or PROCESS.
- b) Each CONDITION should be associated with at least one EVENT or PROCESS.
- c) Each CONDITION should have a TRUE WHILE or FALSE WHILE statement.

.....

VII) SYSTEM PROPERTIES
- a) All KEYWORDS, ATTRIBUTES, SOURCES, SECURITIES and TRACE-KEYS should APPLY to some other URL names.

.....

VIII) PROJECT MANAGEMENT
- a) All PROBLEM-DEFINERS should have a MAILBOX.
- b) All PROBLEM-DEFINERS should be RESPONSIBLE for the description of at least one URL objects.

.....

Table 4.6 URA Reports that may be used by Visually Check for Completeness of the Problem Statement

Analyzer Commands	Completeness Checks
ATTRIBUTE INFORMATION REPORT	VIIa
CONSISTS COMPARISON MATRIX	IIIc, IIId, IIIe, IIIg
CONTENTS REPORT	IIIc, IIId, IIIe, IIIg
DATA PROCESS REPORT	Ic, Id; IVa, IVb, IVc, IVd, IVf
FORMATTED PROBLEM STATEMENT	Ia-Ie; IIa-IId; IIIa-IIIg; IVa-IVf, IVi, IVj; Va-Vk; VIa-VIc; VIIIb
FREQUENCY REPORT	Va, Vb, Vh, Vi
NAME GEN	VIIIa, VIIIb
PICTURE	Ia, Ib, Ic, Id, Ie; IIb, IIC, IId
PROCESS INPUT/OUTPUT	IVb, IVc
PUNCHED COMMENT ENTRIES	IVe, IVj, Vd, Ve, Vg

URA Report	Completeness Checks
CONSISTS COMPARISON MATRIX	- All INPUTS, OUTPUTS, ENTITIES and GROUPS are broken down to ELEMENTS at the lowest level. - All necessary ELEMENTS are defined in the data structure for a particular INPUT, OUTPUT or ENTITY.
CONSISTS MATRIX	- All GROUPS and ELEMENTS belong to higher level data structures. - All SETS broken into INPUTS, or OUTPUTS or ENTITIES[1]
CONTENTS REPORT	- All INPUTS, OUTPUTS, ENTITIES and GROUPS are broken down to ELEMENTS at the lowest level. - All SETS broken into INPUTS, or OUTPUTS ENTITIES
DATA PROCESS REPORT	- All INPUTS RECEIVED by some PROCESS[1] - All INPUTS USED by some PROCESS[1] - All OUTPUTS GENERATED by some PROCESS[1] - All OUTPUTS DERIVED by some PROCESS[1] - All ENTITIES and SETS DERIVED by some PROCESS[1] - All ENTITIES and SETS DERIVED and USED by some PROCESS[1] - All ENTITIES and SETS are UPDATED and USED by some PROCESS[1] - All GROUPS and ELEMENTS are DERIVED or UPDATED or USED by some PROCESS[1] - All PROCESSES USE data and DERIVE or UPDATE data[1] - All PROCESSES which DERIVE data also USE data[1] - All PROCESSES which UPDATE data also USE data[1] - All PROCESSES interact with data in some way[1]
DICTIONARY REPORT	- All names should have a narrative DESCRIPTION and RESPONSIBLE-PROBLEM-DEFINER
DYNAMIC ANALYSIS	- All the dynamic relations for CONDITIONS, EVENTS, PROCESS and INPUTS are broken down to the lowest level.

Table 4.7
Completeness and Consistency Checks Made by URA Reports

---

[1] Computer-aided analysis

EXTENDED PICUTRE	- All SETS are broken into ENTITIES or INPUTS or SETS - All PROCESS interact with data in some manner - All INTERFACES generate INPUTS to the SYSTEM and/or receive OUTPUTS - All OUTPUTS are generated - All INPUTS generated must be used in some manner - All SETS are used, updated or derived - All INPUT, OUTPUT, ENTITY are produced and/or used in some manner - All GROUP, ELEMENT are produced and/or used in some manner - All INPUT, OUTPUT, GROUP, ENTITY are eventually broken down into elements - All GROUP, ELEMENT are contained within some larger data.
IDENTIFIER INFORMATION REPORT	- Determines which ENTITIES have and do not have IDENTIFIERS
INTERVAL CONSISTENCY	- All INTERVALS are broken down into INTERVALS at the lowest level
FORMATTED PROBLEM STATEMENT	- The description of each name can be checked against all possible statements for that name.
FREQUENCY REPORT	- All INPUTS, OUPTPUTS, PROCESSES and EVENTS should have a HAPPENS statement
KWIC INDEX	
NAME GEN	- All names of a particular type (e.g., PROCESS) have been defined for a particular problem statement
NAME LIST	- Names which have synonyms in the real world should have them in the problem statement
PICTURE	[given in Table 2.1b] - All names should be involved in structure and/or information flow of the problem statement[1]

Table 4.7 (Continued)

[1] Computer-aided analysis

ATTRIBUTE REPORT	- All ATTRIBUTES have VALUES - All names have appropriate ATTRIBUTES/ATTRIBUTE VALUES assigned to them
PROCESS INPUT/ OUTPUT	- All PROCESS interact with data in some manner
	- All PROCESSES USE or RECEIVE information[1] - All PROCESSES GENERATE, DERIVE or UPDATE information[1] - All PROCESSES have DESCRIPTION and PROCEDURE statements
PROCESS CHAIN REPORT	- The absence of any EVENT and/or PROCESS in the chain for a given name should be rationalized
STRUCTURE REPORT	- All of the supbarts of PROCESSES, INPUTS, OUTPUTS, INTERFACES and PROCESSORS are broken down into their smallest constituents.

Table 4.7 (Continued)

URA Report	Consistency Checks
CONSISTS COMPARISON MATRIX	- Similarities in data structures of INPUTS OUTPUTS and ENTITIES can be rationalized for a particular problem statement.
CONSISISTS MATRIX	- Determines whether or not the use of the CONSISTS statement in describing structure is consistent.
CONTENTS REPORT	- Determines whether or not the use of the CONSISTS statement in describing structures is consistent.
DATA PROCESS REPORT	- Determines whether or not the use of RECEIVES and GENERATES statements in describing the system flow aspect of the system is consistent. - Determines whether or not the use of USES, UPDATES and DERIVES statement in describing the data derivation aspect of the system is consistent.
DICTIONARY REPORT	- Synonyms and descriptions should apply to each name accurately
DYNAMIC ANALYSIS REPORT	- Determines whether two names participating in two dynamics relationships with each other are consistent
EXTENDED PICUTRE	- Determines whether or not the name the EXTENDED-PICTURE is generated for relates to the structure and information flow aspects of the problem statement correctly
IDENTIFIER INFORMATION REPORT	- Determines whether or not the use of IDENTIFIERS is consistent through the problem statement.
INTERVAL CONSISTENCY REPORT	- Determines whether the CONSISTS structure for INTERVAL is consistent regarding the VALUE of all the different paths in the network
FORMATTED PROBLEM STATEMENT	- Determines whether the complete URL description is accurate for a particular name (e.g., check that the DESCRIPTION matches what is given by other URL statements)

Table 4.7 (Continued)

FREQUENCY REPORT	- Determines whether or not the manner in which frequencies (HAPPENS statement) are assigned is consistent.
KWIC INDEX	- Determines whether or not conventions used in assigning names is consistent
NAME GEN	- Determines whether or not naming is consistent - Determines whether or not name types have been assigned correctly.
NAME LIST	- Determines whether or not naming is consistent - Determines whether or not name types have been assigned correctly - Determines whether or not SYNONYMS have been assigned correctly
PICTURE	- Determines whether or not the name the PICTURE is generated for relates to the structure and information flow aspects of the problem statement correctly
ATTRIBUTE REPORT	- Determines whether or not the conventions of assigning ATTRIBUTES is consistent
PROCESS INPUT/ OUTPUT	- Determines whether or not the manner in which PROCESSES are described is consistent
PROCESS CHAIN REPORT	- Determines whether or not the name the PROCESS-CHAIN is generated for relates to the dynamic aspects of the problem statement correctly.

Table 4.7 (Continued)

5. LOGICAL DESIGN OF INFORMATION PROCESSING SYSTEMS

(for designers)

5.1 Content, Form and Method of System Representation

Before the analyst starts to define the target information processing system, he should determine what conceptual model he is going to use to define and then to describe it. The final form of the logical design is invariably some kind of documentation. Much of the difference of opinion regarding the documentation subject arises out of the failure to distinguish between "content" of documentation and the "form" in which it is presented. Content refers to what has to be described, what has to be said, how detailed the description must be, etc. Form on the other hand refers to the manner in which the information is presented. In many cases the same content can be presented or represented in more than one form.

The information that is documented, i.e., presented in some form, must somehow be obtained or produced. The term method will be used to describe the manner in which the information supports the creative process of the system development and its documention in content and form. URL is a very flexible and comprehensive language. Most situations can be represented or expressed in URL in more than one way; each of which is syntactically correct. However, the different representations may imply different semantics which may or may not be what the analyst intended. This section describes a number of situations in which alternative methods of expression are possible and outlines the implications of some different strategies for analyzing and expressing the logical system design.

Systems are composed of parts which interact and are interdependent. Systems may naturally exist, or they may be man-made. What is defined to be in a system, i.e., its scope, is essentially arbitrary, at least until an appropriate criterion is given. The definer decides what the boundaries of his system are; anything that is not within the system is outside or in the environment.

The system may frequently be divided into subsystems and other components. These are further subdivided and so on until no further subdivision is desired. The lowest level of components in any decomposition will be referred to as elements in this guidebook. The parts of the system interact through interconnections or interfaces.

There are usually many different ways in which a given system can be divided into subparts or elements. There are a number of ways in which a given number of elements can be interconnected to form systems. The discussion below will impose a methodology for the analysis of information systems.

The subparts of a system (subsystems and components) are the

composition of a system. The structure of a system encompasses the interactions or interrelationships of the subparts. The process of subdividing a system is called decomposition or factoring. The process of combining smaller systems into larger systems is called synthesis.

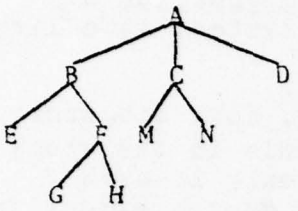
Some interrelations, under certain conditions, form hierarchical or tree structures, which play an important role in the study of systems. A structure is hierarchical if and only if each subsystem is a part of one and only one other system except for the top system. The head, or root-system, is not part of any other system. This structure can be represented in various forms as shown in Figure 5.1.

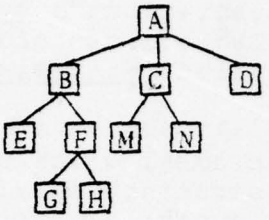
Hierarchical structures play a central role because many systems exhibit this characteristic. Some argue that the fundamental way in which human beings deal with complexity is by hierarchical subdivision, Simon (1962).

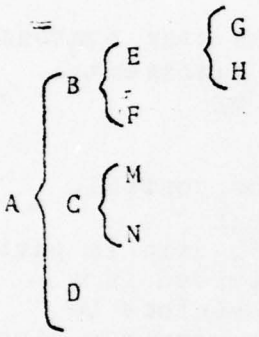
Systems are frequently organized hierarchically for control. Many organizations, for example, have a hierarchical organization structure in which each organizational unit is part of another organizational unit and is itself subdivided into other organizational units. This structure may therefore be represented by an organizational chart. The same structure also represents some information flow, the information that is necessary to issue instruction (downwards) and receive reports (upwards). This organizational structure does not represent material flow, since only the lowest level units are involved in actual operation on material. Thus the flow of information concerning materials does not follow the structure lines. In order to understand what a given system component does in detail, it is helpful to separate it into its constituent parts or subsystems. However, to understand what a system does, it is necessary to know, not only what each subsystem does, but how they interrelate.

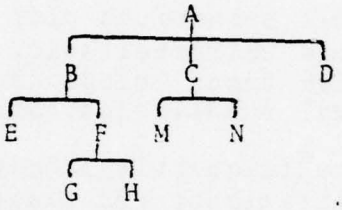
Section 5.2 presents a system description model that will serve as the basis for succeeding sections. Section 5.3 outlines the multi-dimensional character of the structure of the information processing system.

Figure 5.1 REPRESENTATIONS OF TREE STRUCTURES










```
1          A
  2          B
    3          E
    3          F
      4          G
      4          H
  2          C
    3          M
    3          N
  2          D
```

```
1.0        A
1.1          B
1.1.1          E
1.1.2          F
1.1.2.1          G
1.1.2.2          H
1.2          C
1.2.1          M
1.2.2          N
1.3          D
```

```
(A,B)
(A,C)
(A,D)
(B,E)
(B,F)
(C,M)
(C,N)
(F,G)
(F,H)
```

	A	B	C	D	E	F	G	H	M	N
A		1	1	1						
B					1	1				
C									1	1
D										
E										
F							1	1		
G										
H										
M										
N										

5.2 General Conceptual Model of Systems Description

As the basis for comparing a number of existing and proposed methods for developing and documenting systems, one must start with a system's description model comprehensive enough to encompass all the variety and complexity of information processing systems, and one that has been proven to be operationally complete. In order not to be biased for or against any of the methods described below, it is necessary to use a model that is defined independently of the methodologies. It is widely recognized and generally accepted that such a "model" should be based on systems theory. For example, Couger and Knapp (1974) begin their survey of system analysis techniques with a section on "The Systems Perspectives." Katzan (1976) begins his book on System Design and Documentation with chapter containing an "Introduction to System Theory" and a "Taxonomy of System Concepts."

Many models of systems have been proposed and used for various purposes. Among the ones that have been reviewed, and from which ideas have been incorporated into the model outlined below, are the following:

1) General systems theory models described, for example, by Bertalanffy(1972).
2) Models of organizations as systems as described by Simon (1962), Anthony (1965), Emery (1969).
3) Systems engineering models as described by Hall (1962) and Machol (1965).
4) The industrial engineering approach to system development operation and maintenance, which has, in numerous examples, convincingly proven its practical capability to deal with very complex real-world problems, without falling into redundant formal considerations. An example is the model proposed by Nadler (1967,1969) modified in content, definition and interpretation by Gackowski (1969,1974,1976).
5) Simulation models including those of Forrester (1961) and other models implied by various simulation languages (SIMSCRIPT, SIMULA, etc.).

The complete description of any existing or hypothetical system should cover the topics (rows) and aspects (columns) summarized in Table 5.2 independent of the size of the system.

A system is considered to be any entity, consisting of parts or elements and the interactions among them, that produces a measurable effect in spite of the interference (within determined limits) of its environment. This definition therefore excludes from consideration systems which do not do anything of interest or are only abstract constructs. In an active, objective-driven organizational environment, a system must be defined in relation to the effect it produces.

Therefore, the first and major topic in the description model is the objective, mission, effect or purpose of the system (row 1 in Table 5.2). For man-made systems it is the reason for the development, operation and maintenance of the described system.

The system's functions (see row 2 in Table 5.2) delineates the type of system activities from which the effect is obtained, and thus qualitatively distinguishes the type of the system. If these functions are changed, a different type of system should be described. For instance, improved profitability of an organization can be achieved by more advertising, manufacturing a different product, by providing information on more beneficial investment opportunities, etc. If the main function of a system is information processing then it is called an information processing system.

The next most important topic in the description of systems is the outputs (see row 3 in Table 5.2) it produces, for example, materials, goods, services, energy, information, persons like graduate students from a university, patients from hospitals, etc. Of a particular interest are the system's main outputs, which directly contribute to achieving the system's main effect. When the acting system's main output is information (always in the form of some kind of data, messages or signals), it is an information system.

In principal there is no direct connection between information processing and producing information as output of a system. An engine during a test run is a classic example of an information system which does not process information at all. It is run exclusively in order to produce information about its behavior but not to produce power. An inter-continental missile defense system is a good example of the opposite. This type of system processes only information gathered by warning radar systems but the main outputs are not warning information but the guided intercepting missiles. Only the missiles contribute to the main objective (effect) of an effective active defense system to destroy the ICBMs before they reach the targets. Otherwise it would be only a warning system, which would be both an information processing system and an information system. Of course, there also exists other possible outputs, and some of them may even be very undesirable.

The next aspect of the system is the method and processes or mode of operation by which the main outputs are obtained (called basic and all others, the auxiliary ones which contribute only to the accomplishment of the basic process (see row 4 in Table 5.2). In man-made systems, the auxiliary methods and processes are those that contribute to the accomplishment of the basic process in compliance with all additional constraints and requirements imposed on the system's operation. For instance, all checks made to assure the correctness of output data, uninterrupted operation of the system, and privacy protection for sensitive data are auxiliary methods. This topic covers all the dynamic interactions between system parts such as the flow, of control system data derivation, and system's management in information processing systems.

The method of producing the outputs and performing the system's function calls for some necessary inputs (see row 5 in Table 5.2) of similar variety as specified for outputs. For information processing systems, the main inputs are usually all

kinds of data carrying information into the system.

The method of transforming inputs into outputs calls for some means (facilities, equipment, tools, space, financial funds and other resources) (see row 6 in Table 5.2) with which the system is equipped in order to perform the necessary processes. Contemporary information processing systems in which many functions (not necessarily all) are performed by a computer are called computerized or computer-based information processing systems. Since this paper deals mainly with the development and documenting of such systems, the term information processing system refers to computer-based information processing system.

Any man-made, not fully automated, system needs some personnel for its operation and maintenance, as well, of course, as personnel for its development. The system personnel does not include persons who are the objects of action of the system such as university students, hospital patients, persons trained by a computerized instruction system. Information-processing-systems personnel encompasses:

- during the operational stage of its lifecycle:

  system managers, system operators, hardware-software operators, data base administrators, system and software librarians, hardware and communication facilities, maintenance engineers,

- during the development stage of its life cycle:

  project leaders, system analysts, system architects, data base designers, software designers, module programmers, project librarians.

Systems cannot be described in isolation of their environment. A system is capable of handling and managing only certain kinds of environments, while resisting the interference. The characteristics of the environment is the last topic of the system description model (see row 8 in Table 5.2) covering such factors as: dust, temperature, humidity, vibrations, noise, chemical active atmosphere, danger of violations of data privacy, system's security, and so on. Man-made systems usually operate in an economics-driven environment. This fact makes it necessary to include all aspects of the system's economics.

In order to make the system description model complete, each of the above mentioned topics must cover the following aspects:

- The appearance form of objects covered by each of the above specified topics, its composition (qualitatively by type, and quantitatively by instances of each type) and its composition or decomposition structure (see column 1 in Table 5.2).

- The intensity of the above in time (quantity per time unit for stable phenomena, first derivative of quantity function of time for continuous phenomena, frequency or distribution in time for

discrete phenomena) (see column 2 in table 5.2). For instance the number of transactions or inquiries per time interval, their distribution in time to deduct the pickloads, etc.

- Their tendency to change with time (second derivative of quantity function of time for continuous phenomena and qualitative changes). For instance, is the system in a state of decline or growth, or should it be able to cope with an increasing amount of transaction processing (see column 3 in Table 5.2)?

The system's description model given above, is comprehensive enough to cover all important objects, relations, factors and aspects which are necessary to completely document a system. This does not mean that always, for all purposes, the whole description should be carried to the lowest possible level of detail. Rather, this description model should be considered as a reference model which when used as a check list, eliminates the possibility that important aspects are omitted. For some purposes the description of only some particular aspect of the system may be needed. However, even if a complete description is necessary, the general rule stating what should and what should not be included in the description of a specific system is as follows:

> Only those objects, relations, factors and aspects should be entered into the description which causes an impact on the main effect of the system and/or on the assessment of its quality, efficiency, or effectiveness (according to a predetermined evaluation criterion) of a magnitude exceeding the predetermined tolerance limits.

In general, any system can be considered as being represented by a complex graph which includes all objects and relations necessary according to the above rule. The complete graph can be stored in a computerized data base and can be presented graphically in the form of a network, in matrix, or in set form.

Table 5.2 OUTLINE OF THE SYSTEM DESCRIPTION MODEL

Aspects of Topics / Description Topics	Form of Appearance	Intensity (in time)	Tendencies (time trends)	Others
	1	2	3	4
1. Effect				
2. Main function				
3. Outputs				
4. Method, process				
5. Inputs				
6. Means				
7. Personnel				
8. Environment				

5.3 Information Processing Systems Description Model

Man-made systems, in particular, should be defined first of all with respect to the main effect that they are expected to produce, and, secondly, to their components and the functions they perform. Accordingly, the structure of any information processing system consists of all the relations between the system's parts, the system itself, and its environment, which produces an impact (of magnitude exceeding the pre-determined tolerance) on the expected system's effect on the organization it serves.

In URL the documentation content can be expressed by means of several types of language objects (such as nouns, verbs, adjectives, etc.):

a) To designate precisely:
- objects actively or passively interfacing with the system on its boundaries, such as organizational units, software units, etc.
- data used, usually system's outputs.
- intermediate and auxiliary data, often system's data base.
- processes performed and procedures followed.
- facilities or means used.
(all at various levels of detail or generality).

b) To distinguish them uniquely by some naming convention.

c) To relate them with some kinds of properties.

d) To describe their size and volume.

e) To relate them mutually by a set of typical and specially-defined relations in a consistent system structure in all its aspects (subdivision, flow, etc.).

f) To describe their dynamic behavior in relation to some conditions and events related to the time flow.

g) To describe the resource consumption by them during the development and operation of the system.

h) To facilitate the comprehension of the system, and the management of its development, operation and maintenance.

To adequately represent organizational systems and information processing systems one must provide in the description for a number of different kinds of objects and relations. A conceptual model of the documentation of information processing systems include the following types of aspects:

1) The system's effects, especially the expected effect of operating the information processing system on the organization which it serves, expressed as a function of the system's output data. At present the most convenient object type for this purpose is a MEMO with an adequate DESCRIPTION (see example in

section 3.3)

2) The system's flow which consists of all interactions between the information processing system and its environment including all constraints and requirements imposed on it.

The system's environment itself can be described by INTERFACES; its interactions with the system via INPUTS and OUTPUTS and the URL relations RECEIVES/RECEIVED BY and GENERATES/GENERATED BY. Requirements concerning the INPUTS and OUTPUTS in respect to
- their content can be stated via SUBPARTS ARE and CONSISTS OF relations,
- their occurrence via HAPPENS relation,
- other aspects via KEYWORDS, ATTRIBUTES with ATTRIBUTE-VALUES and DESCRIPTION.

3) The hirearchical structure of all kinds of systems objects especially the data which the system acts upon: SETS, ENTITIES, GROUPS, ELEMENTS; PROCESSES which are PERFORMED by various types of PROCESSORS (organizational units, management and staff positions, computing equipment, etc). All data objects can be decomposed in URL via the CONSISTS OF (except ELEMENTS) and CONTAINED IN relations. In addition, INPUTS and OUTPUTS can be decomposed via the PART OF/SUBPARTS ARE relations, and SETS via SUBSET OF/SUBSETS ARE relations. PROCESSES and PROCESSORS can be decomposed only via PART OF/SUBPARTS ARE relations.

4) The system's data derivation in which OUTPUTS and all intermediate results in the form of SETS, ENTITIES, GROUPS and ELEMENTS can be precisely expressed as a function of INPUTS or other intermediate data objects via three types of URL data derivation relations:

- DERIVED BY process-name USING input-, set-, entity-, group- or element-name,
- USED BY process-name TO DERIVE output-, set-, entity-, group-, element-name (not applicable for OUTPUTS)
- UPDATED BY process-name using input-, set-, entity-, group-, element-name (not applicable for OUTPUTS),
- or the respective complementary URL relations to those mentioned above and used with PROCESS.

5) The system's size, especially the volume of processing, and other properties of system objects which have any impact on the expected system's effect on the organization it serves can be expressed in URL via SYSTEM-PARAMETER names or values with the CONSISTS OF, CARDINALITY, VOLATILITY, CONNECTIVITY and HAPPENS statement or attribute-names with appropriate attribute values.

6) The system's dynamics which describe the relations between all objects in the system and time can be expressed in URL by using the event-, condition-, and interval-object-type-names via the TRIGGERS, ON INCEPTION OF, ON TERMINATION OF relations and the WHEN condition-name BECOMES either TRUE or FALSE statements.

7) The system's facilities in relation to flow, data derivation, dynamics and size can be described in URL by means of defining

different kinds of PROCESSORS involved and related via PERFORMS relation with appropriate process-names.

8) The system's management during the development and operation stages can be supported by using names of URL object types such as PROBLEM-DEFINERS, SOURCES, MEMOS, MAILBOXES.

9) The system's economics. The CONSUME statement for all PROCESSORS, PERFORMED and RESOURCE-USAGE with RESOURCE-USAGE-PARAMETER-VALUES statements for all PROCESSES, and MEASURED IN statements for RESOURCES, form a complete basis for a thorough analysis of system operational costs. However the comparison of costs and expected benefits and respective calculation of effectiveness indicator have to be performed manually at this time.

The above mentioned aspects constitute different dimensions of information processing system description. These are not necessarily all of the aspects, but they are the most important.

5.4 Ordering of Design Activities and Description of the System

Defining the order in which to ask questions about the target system until the specification is complete is not a trivial task. It is the major subject of most system development methodology studies. There is no common agreement on the order in which to perform the system development, and thus its description.

While the approach that should be used in a particular organization will normally be selected by management, it may be helpful to review a few alternatives.

Information processing systems and software system development certainly are separate branches of systems engineering, but not so much different that it could not apply and profit from some advances in its other branches and the more general achievements in systems engineering methodology. However, it seems they are widely ignored. Still, there are some rules which have been proved to be generally valid, which never compromise but statistically considerably improve the effectiveness of the system development process.

5.4.1 The Flow-up Engineering Approach

According to the present state of system engineering, any system development method should facilitate and increase the effectiveness of carrying out the following important design steps:

1) A thorough analysis followed by the determination of what (in content, form, quantity, distribution in time) should be the system's main outputs, that is, those which directly and quantitatively contribute to the attainment of the system's objective, and where and how the outputs should be effectively

used. This can be done by a systematic analysis of the decision situations of the potential users of the system's outputs, and thus determining the value of information conveyed to them.

2) A thorough analysis of the available methods of obtaining (if they are not already available) the previously determined outputs, and the determination of the preferred ones in the light of the system evaluation criterion.

3) If the output data must be computed (they are not available in ready form) an analysis is necessary to find out where and at what cost the input data required by various methods could be acquired. Again, the evaluation of the conceivable combinations of methods and input data (always by the same criterion) to produce the desired outputs should be used to determine (at the next step of approximation) what outputs should be derived from inputs, and the particular procedures necessary.

The major outcome of these first three design steps is a fairly accurate logical specification of the procedures necessary to derive the determined output data from the determined input data. At this stage, all of the system's constituents are specified only in logical terms. The final specification of the system's data and procedures must additionally take into consideration such factors as the means to be used, the staff to be employed and any applicable environmental circumstances. However, these factors can also call for some modification of the logical specification of system's data and procedure.

4) The analysis of available data processing facilities, their impact on the cost performance factor for the determined processing methods (not yet procedures), and the amount of input-output data in confrontation with other user's requirements should indicate how the data processing should be performed, that is, which part of the processing should be computerized, and which part should be done in other ways, including manually. Only at this point is it feasible to start the development of the necessary user dedicated application software system.

5) Having determined the methods and means of processing, the system designer is now prepared to analyze all human factors related to the information processing system development and operation. These factors included the kind of staff that will be necessary in order to operate the system's hardware and software, and to perform all computerized and non-computerized (manual) procedures of transforming the input data into output data.

6) The specification of the means and methods embedded in the system design that are responses to various aspects of environmental circumstances within which the system is intended to be operated, for instance: privacy, security and auditing regulations.

The order of the design steps presented above has not been set arbitrarily. Of course, there are almost no physical or logical

constraints that prevent performing the design process in any order, but not without adverse impact on its effectiveness.

The order "input-process-output" which seems to be natural because it is flow oriented, appears to be the correct order of system design but it severely compromises effectiveness. A system designer following this sequence often starts with specifying the input data. This way, he subconciously limits the potential processing method alternatives only to those dealing with the previously specified input data. In further consequence, he also limits the scope of output data alternatives to be considered. What often results is the design of a system that fails to meet its main objective. This maybe does not much harm in case of simple data processing system but can have serious adverse impact on decision supporting information processing systems. Most descriptions do not go beyond the output data and considers them at the very end of their design efforts. They substitute the achievement of measurable effects within the organization by sophisticated forms of output data.

Similarly, the system designer, who first makes up his mind upon computing facilities is later heavily restricted in data design and process design. Again, the system designer, who since the very beginning is concerned about environmental constraints before he has elaborated a solution meeting the main objective, can hardly ever find a satisfactory solution, because he is over-constrained in his search for adequate solutions.

Assuming initially that there are no severe constraints imposed on the system being designed, the order of the system design will be summarized. The system is defined in relation to the main effect. The main effect is the major determinant of the system's outputs which contribute to produce this effect; the main outputs - for the methods used to obtain them; the main outputs and methods of deriving them determine the necessary inputs, and the chosen method of processing determines the necessary means of processing. The latter determines the staff necessary to operate the system. Eventually, the entire specification of the system must be adjusted to the requirements and constraints imposed on the system by its environment. Any attempt to change this order of design steps leads to unnecessary constraints on the designer's freedom to exert his creativeness, which inevitably results is a decline of effectiveness of the design process.

5.4.2 The Top-Down Approach

The top-down approach dictates that the entire PROCESS performed by the system be divided functionally into a few large pieces. Inputs to and outputs from these pieces are described. Each of the pieces is then regarded as a function to be subdivided similarly. The division process is continued until the smallest pieces can no longer be partitioned in a logical fashion. The partitioning process may be accomplished easily by means of URL PROCESSES and the SUBPARTS statements. An adaptation of this

approach is illustrated in the URL/URA Training Example, ISDOS Working Paper No. 156.

5.4.3 The Bottom-Up Approach

The bottom-up approach is normally used when it is expected that the system will be constructed from a number of predefined, and pre-existing components. The components are grouped into larger packages, which are in turn grouped among themselves to perform more complex functions. The grouping process is continued until the entire system is constructed.

5.4.4 The Boundary Approach

A URA data base contains the description of one system. Every system has a boundary and a system description that may be thought of as consisting of two parts:

- the specification of what crosses the boundary,
- the specification of what goes on inside the system.

Alternative strategies are possible with regard to the order in which these parts are specified. One possiblity is to delineate the system boundary first. The second is to describe the "interior" of the system without identifying the boundary (see Figure 5.3).

A) Specifying the boundaries first

A distinct boundary is obtained when the INTERFACES are defined and their communication with the system is specified by naming the INPUTS and OUTPUTS. It is assumed here that the INPUTS enter the system and the OUTPUTS leave the system in some physical form containing data values. The constraint in URL is that an INPUT can only come from an INTERFACE and OUTPUTS may only go to an INTERFACE. Inside the system a number of PROCESSES may be named, each one of which uses data from the available sources - INPUTS, ENTITIES, or SETS, or from unspecified sources - GROUPS and ELEMENTS, to derive and update data. A PROCESS may USE data from any of these sources or data that are DERIVED from any PROCESS; and similarly, data DERIVED by one PROCESS may be USED by any number of other PROCESSES.

One benefit of this approach is that the problem statement can be checked for completeness, e.g., that each INPUT is GENERATED by some INTERFACE and RECEIVED by at least one PROCESS, and that each OUTPUT is GENERATED by some PROCESS and RECEIVED by at least one INTERFACE. Another benefit is that the description of the INPUTS and OUTPUTS can be agreed to by the relevant INTERFACE.

B) Specifying the interior first

In some cases, it may be desirable to delay specifying the

system boundary until the interior of the system has been described. This can be done by not identifying any INPUTS and OUTPUTS, but instead, defining those PROCESSES that USE and DERIVE data and defining data in terms of ENTITIES, SETS, GROUPS and ELEMENTS. What would be an INTERFACE in the previous case, can now be identified as a PROCESS, can use data from any source-derived data that can be used by any other PROCESS.

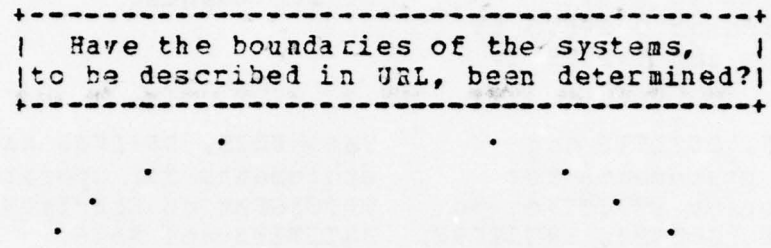
The advantage of this approach is that any of the objects, e.g., PROCESS, can both USE and DERIVE data, and that a given collection of data identified as an ENTITY can be both USED by a PROCESS and DERIVED by a PROCESS (in addition of course to being UPDATED.)

However, there is also a serious disadvantage of this approach, that the analyst pays too much attention to the inside details of the system at the expense of its duty to please the outside user.

Other examples of ordering the analysis, design and the description process for data bases are outlined in Figure 5.4.

Several other approaches are possible. URL is sufficiently flexible to allow its users many options in the methodology used.

Figure 5.3 Defining Objects and Relationships in Relation to System Boundaries



YES	NO
Define as:	
-INPUTS the types of objects containing data which enter the system, i.e., cross the boundary	
-OUTPUTS the types of object containing data which leave the system, i.e., cross the boundary.	{Define all objects containing } {data as ENTITIES or GROUPS. } {Use ENTITIES whenever there may be} {many instances of the type of } {object in the target system. }
Define as ENTITIES all objects containing data which reside permanently inside the system.	
Define as PROCESSes the types of objects which operate on data inside the system.	Same
Define as INTERFACES the types of objects, outside the system, that send INPUTS to the system or receive OUTPUTS from it.	Define as PROCESSes all objects which operate on data. Therefore, real world objects such as people, organizational units, information processing systems, etc. will all be defined as PROCESSes.
Use RECEIVES/RECEIVED BY statements to show flow of INPUTS to PROCESSes	Do not use RECEIVES/RECEIVED BY statements.

and OUTPUTS to INTERFACES.

Use GENERATES/GENERATED BY statements to show production of OUTPUTS by PROCESSes and INPUTS by INTERFACES.	Do not use GENERATES/GENERATED BY statements.
Use USES, DERIVES and UPDATES statements for operation by PROCESSes on ELEMENTS, GROUPS, ENTITIES, INPUTS, OUTPUTS and SETS.	Uses USES, DERIVES and UPDATES statements for operation by PROCESSes on ELEMENTS, GROUPS, ENTITIES and SETS.
Define as SETS all collections of (multiple instances of) INPUTS, OUTPUTS, ENTITIES.	Define as SETS all collections of (multiple instances of) ENTITIES.

In all other respects, the process of describing the system is exactly the same in both cases.

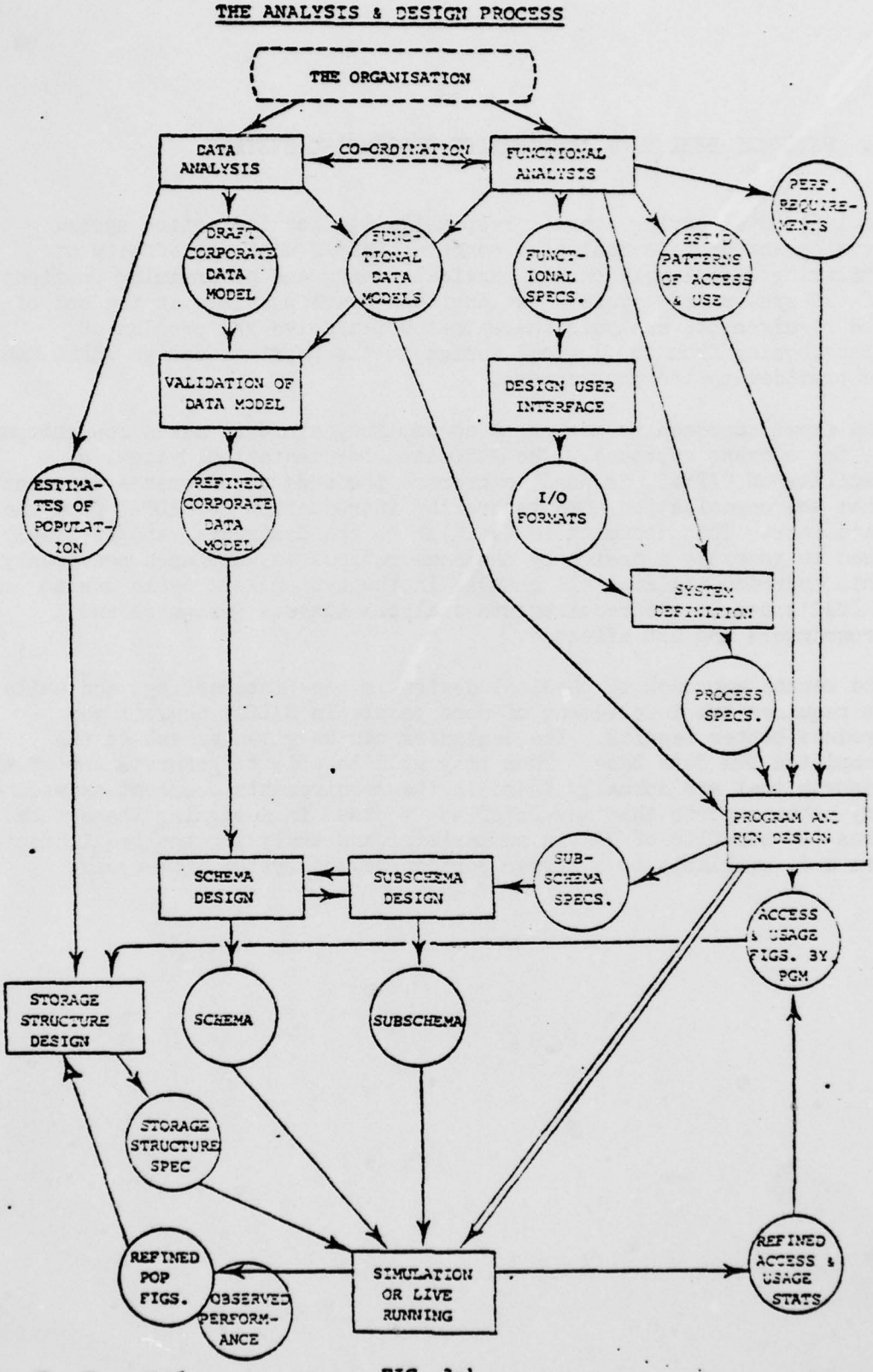


FIG. 3-1

TOZER, E. E., "DATABASE SYSTEMS ANALYSIS AND DESIGN," LECTURE NOTES IN COMPUTER SCIENCE, ECI CONFERENCE 1976, PROCEEDINGS OF THE FIRST CONFERENCE ON THE EUROPEAN COOPERATION IN INFORMATICS, APRIL 1976, pp. 193-224.

FIGURE 5.4 ANALYSIS AND DESIGN PROCESS OF DATA BASES

6. PHYSICAL DESIGN OF INFORMATION PROCESSING SYSTEMS

An individual having overall responsibility for the entire system development process must also concern himself with the effects of utilizing CADSAT data on the physical design and programming portions of the system life cycle. Two approaches are possible at the end of the requirements analysis phase that would solve the problem of transforming from the logical design to the physical design which must be provided to the programmers.

The first approach is easier to accomplish, since it makes few changes to the current approach. The Automated Documentation System, a facility of CADSAT, is used to prepare the same requirements document that the organization used before the introduction of CADSAT from the data base. This document is familiar to the designers, and is simply used to generate a design by the same methods as were used previously. This approach isolates all changes in the system life cycle due to CADSAT's use in the requirements analysis stage. Designers and programmers are not affected.

The second approach to physical design is more interesting, and while it requires the involvement of more people in CADSAT use, it may produce better results. The designers can be given access to the completed URA data base. Then they will be able to generate any of the reports that are normally found in the requirements document as well as any other reports that may be of use to them in designing the system. Thus the benefits of URA in summarizing and analyzing the requirements are made available to a larger portion of the system life cycle.

7. COMPUTER-ASSISTED GENERATION OF SYSTEM DOCUMENTATION

(for project leaders, systems analysts and designers)

Documentation of information processing systems is used for many purposes and by different users during the sytem's life cycle.

7.1 Differentiation of Documentation with System Life Cycle

In the early application of computers, documentation was undifferentiated and one report or one document had to serve all individuals involved at all stages of the life cycle. Gradually it has become clear that several different (though related) systems are of interest and should be documented. The first system to be described is the present form of the organization in which a new or proposed information processing system is to be embedded. The next system is the proposed information processing system as seen from the organization in which it is embedded. This now is frequently called the functional description or the logical view. Next is the physical information processing system. This includes a number of components one of which is software. Software itself may exist in several different stages: functional specification, source modules, "compiled" modules, and executable code (modules which have been linked and loaded).

As documentation has evolved, the content and form has been differentiated to suit the needs at the different stages. However, the transformation from one form to another has, up to now, had to be undertaken manually except for source code to object code. Current developments are directed toward providing manual aids for verifying and validating these transformations and eventually for performing these operations by computer.

7.2 Differentiation of Documentation for Different Users

Along with the differentiation of the documentation to represent different views at different stages of the life cycle, it has also been recognized that individuals associated with systems development need a variety of documents of dissimilar content and format for their particular purposes. Most system development procedures and documentation standards now specify that particular documents be developed for the various classes of users.

The following classes of users of documentation may be identified:

- Individuals who perform some task on the basis of previous efforts or other efforts in order to continue it but it must be documented.
- Reviewers who must review documentation for correctness, completeness, etc.
- Implementors who use documentation as the specification for

their own tasks.
- Approvers who must approve the documentation.
- Documentors who must prepare and maintain the actual documentation.
- Maintainers of the system which is documented. They use documentation to locate and correct errors or make changes for other reasons.
- Readers who are interested in obtaining a general understanding of the system.
- Computer which can assist the process of system development and perform all the automated procedures.

The complete documentation of a system includes the executable object code. Therefore, at least this part of the documentation must be in a form acceptable to the computer. However, since the computer is increasingly being used to store and analyze other documentation and produce, on request, hard-copy documentation for human use, one of the factors to be used in evaluating the forms of the presentation and methods is their suitability for use in a computer-aided systems development. The trend to produce documentation for specific groups of users is being continued particularly for computer-aided systems. The documentation with appropriate content and form can be more readily produced for a particular user.

7.3 Characteristics of Good Documentation

This section outlines some major attributes of good documentation and indicates how an analyst may use URL/URA to achieve them. Figure 7.1 outlines some characteristics of present manual documentation and some desirable characteristics that are achievable with computer-aided documentation.

Present Manual Documentation	Desirable Characteristics of Computer-Aided Documentation
Ambiguous	Precise
Inconsistent	Consistent
Incomplete	Complete
Incorrect	Correct
Difficult to Analyze and Evaluate	Computer can be used to Analyze and Evaluate
Hard to Modify	Computer can be used to update
Format Varying	Fixed Format
Few Standards	Enforced Standards

Figure 7.1 Characteristics of Documentation

To achieve the potential benefits of computer-aided documentation requires:

- a formal language which permits relationships to be precisely defined,

- a computer program which provides a method for enforcing correct use of the formal language,
- good procedures to be followed by the analyst.

The last of these is important since no matter how good the language and the computer software, the benefits will never be attained unless the methods are used properly.

Understandability

Documentation with this characteristic is in an easy-to-read format and is presented at a level general enough so that persons, no matter what their background, can read and comprehend the material. Reports can be generated from the problem statement in several common formats, (e.g., flow diagrams and matrices) and at different levels of detail. For example, it is often desirable to initially present high level objects. Subsequent reports will present more and more detail about these objects until everything is described in terms of their lowest level constituents. The analyst can choose the ordering and content of the reports.

Preciseness

Documentation with this quality must have all relevant terminology explicitly defined so that information presented cannot be misinterpreted. A computer interpreted language must have an accurately defined syntax. The reserved words in the syntax of URL are used to describe different system objects and the relationships between the objects. Definitions of all reserved words allowed in the syntax are fixed so that all relationships presented in the documentation (URA reports) are exactly the same as those initially specified by the analyst i.e., there can only be one interpretation of the information.

Consistency

Documentation which is "consistent" presents all the material in proper context and does not have statements that are conflicting or contradictory. The context in which a particular object is to be used is defined by the user via URL statements that will be stored in the URA data base. Any attempts to use the previously defined object in a conflicting context will result in an error diagnostic. Therefore, use of URA maintains consistency throughout the documentation.

Completeness

To be "complete" the documentation must present the material in sufficient detail so that no reference to outside sources is needed for a thorough understanding of the subject matter. Every necessary piece of information must be available and no

relationship must be left dangling.

URL allows a number of relationships and objects to be defined to describe an Information Processing System. The URL statements offered provide a thorough outline of what should be incorporated into the documentation of an information processing system. The statements in URL facilitate the enforcement of completeness.

Correctness

To be "correct", the analyst must insure that all relationships specified in the documentation are valid, and that all of the information recorded is true. The syntax rules enforced by URA insure that all relationships in the documentation are valid. Though it is impossible to know whether the information recorded is true or not, many of the reports available can present the information in a format that assists the analyst with his search for errors e.g., misspellings, incorrect narrative descriptions, etc..

Analyzability

Documentation which is analyzable must be organizated in such a way that any information not explicitly stated in it must be easily derived through some procedure. Since all URL statements are stored in a data base, all data can be efficiently accessed and presented in the form of a URA report. In addition, any new developments in analyzing the information (e.g., Cost/Benefit Analysis, etc.) can be incorporated into the existing URA package.

Ease of Modification

Documentation which is easy to modify must have sufficient indexing facilities so that all occurrences of a given item in the documentation may be referenced if and when a change to the item is required. Because the information used in deriving URA reports is contained within the URA data base, any modifications to the data base will be reflected in reports produced after the change. URA offers several commands to modify the data base. Any reports generated after the modification will be up-to-date.

Attractiveness

A report should be an attractive, legible document. Ideally, it should be typed in a large legible font on bond paper with figures drawn in color. In practice this is usually not feasible, particularly since the report should be produced, as much as possible, directly as computer output. However, the document should be as attractive as possible.

URA outputs are designed to print an 8 1/2 by 11 page so that

the computer-printed output is the same size as pages prepared manually.

Computer outputs frequently have two features, other than awkward size, which is undesirable: upper case letters only and poor reproducibility. Computer output is difficult to read when only upper case letters are available. Whenever possible, printers with both upper and lower case should be used. Even when lower case is available, the type font is frequently hard to decipher. Since documents must usually be produced in multiple copies, the computer printed master must be of high quality in order to produce legible copies. The Figure 7.2 presents the standard format for Analyzer Reports.

Figure 7.2 Standard Format for Analyzer Reports

Analyzer Version producing the report[1,2] Date and Time produced[1,2]
Page Number[1,2]

Name of the Information Processing System[2]

Name of the Report

Parameters in effect in the production and printing of the report.[2]

Line numbers[2] Body of the report[3,4]

```
---          ------------------
---          .                .
---          .                .
---          .                .
---          .                .
---          .                .
---          ------------------
End of report
```

[1] This information is supplied by the operating system and appears at the top of each page.

[2] Optional.

[3] URL reserved words are printed in upper case and the system description is printed in lower case.

[4] The body of the report may contain information not initially expressed in URL; this is enclosed in /* and */.

7.4 The Automated Documentation System

The Automated Documentation System is a separate, stand-alone facility that can be used to produce a complete requirements document from a URA data base. The final document consists of URA reports and additional text, formatted as desired. A complete description of this facility and instructions for its use are found in Part V of the URA User's Manual.

The way in which the physical design should be produced from the requirements will depend heavily on how the process of physical design has been carried out in the organization in the past. This will frequently determine the methods to be used.

In general, the designer who is trying to determine the structure of the system, in terms of modules and the functions which they must accomplish, should concentrate on PROCESSES. If each PROCESS at the lowest level is to become a module, a Formatted Problem Statement for the PROCESS, which shows all information concerning the PROCESS that is stored in the data base, may be of use to both designers and programmers. On the other hand, there are many reasons why modules may not correspond to PROCESSES on a one-to-one basis. In this case, a general view of the system, perhaps as shown by the Structure report, may be of help in determining the decomposition into modules.

7.5 URA Reports in System Life Cycle

When URL is used to describe a system, all of the description is placed in a URA data base. This data base may be used during the life of the system for various purposes:

1) During requirements development activities (while data is being added and modified by analysts and project leaders) to:

   - verify additions and changes,
   - analyze system,
   - determine what more is needed.

2) After requirements are completed, for:

   - final documentation (functional requirements report),
   - understanding,
   - specifications for succeeding activities.

3) During systems maintenance, in order to:

   - determine effect of changes,
   - change evaluation and authorization procedure,
   - change implementation.

7.6 System Definition Report Using URA Data Base

This section is concerned with the second use, namely, the production of a complete, human-readable statement of the system described in the data base. Usually, such a document will be necessary after the requirements definition is complete in order to receive approval of the proposed system from its potential users, from management, and from others.

It may be necessary to produce this document at other times, but whenever it is produced, it should be based on all the data in the data base and the material should be presented in such a way that almost anyone can read and understand it, i.e., it should not assume that the reader is familiar with computers, with URL or with URA reports. Therefore, there must be a full explanation of the content of the document and the form in which it is presented.

In cases where the System Definition Report is presented orally, it is possible to vary the order and content as desired during the presentation if appropriate display equipment is available. Another case where the order and content may be varied is where the reader is familiar with URA and its command language and can choose the order in which he studies the system to suit himself. In these cases, the order suggested in this appendix may be varied. For the purposes of this paper it is assumed that the System Definition Report must be produced in hard copy and will be read and evaluated by many individuals with different backgrounds, interests and knowledge of the system, computers and URL/URA.

The suggested order of text for the System Definition Report, when using URL reports is given in Table 7.3.

Data Base Summary shows the number of objects of a particular type in the data base, the number and percentage of these which have synonyms, and the number and percentage which have descriptions for each object type which exists in the data base. Totals over all object types are also given. This information enables management to monitor the progress in describing the requirements. Summaries may be compared with earlier ones to estimate the rate of progress and large efforts in particular areas, such as describing processing requirements or defining data items.

Picture Report gives, for each name specified, a graphical report, or "picture", showing structure, flow, and/or data relationships between that name and other names in the data base. While the same information could be obtained from a Formatted Problem Statement, the Picture Report shows the important relationships at a glance, and, therefore, is a better form of presentation for many purposes.

Structure Report can show the full breakdown into smaller parts of any PROCESS, INPUT, or OUTPUT in the data base. This is useful in verifying that the overall structure of the system is

correct, and that all required processing has been included somewhere in the system. It may also be used to verify that INPUTS and OUTPUTS have been broken down correctly.

Formatted Problem Statement Report, for each name specified, this report will show all information related to that name which is currently stored in the data base. In particular, this report could be used to show information about all names presently stored in the data base. The analyst may generate a Formatted Problem Statement with the ALL-STATEMENTS option for a collection of names and may use this to determine what information about those names is needed, but has not yet been supplied. This may indicate additional questions which must be answered by the user.

Contents Report shows, for all INPUT, OUTPUT, ENTITY, and GROUP names specified, the partitioning into lower level objects specified by the CONSISTS relationships. This is particularly helpful to the user in determining whether inputs to and outputs from the system have been properly described.

Extended Picture Report may show either structure or data flow information in a graphical format. In particular, it may be used to demonstrate the hierarchy of PROCESSES, or to show data flow between PROCESSES.

Process Chain Report presents in a graphical format the sequence of EVENTS and PROCESSES which occur as a result of each name given as input. This report may be used to trace the behavior of the system over time.

Dynamic Analysis Report shows the INPUTS, EVENTS, CONDITIONS, and PROCESSES that influence the activities (PROCESSES) that will be performed and the order in which they are performed in the target system. It also indicates the dependencies of the system dynamics on the condition of the other system objects.

Process Input/Output Report can be used to show, for each PROCESS name specified, the function of that PROCESS and the data items with which it interacts.

Frequency Report shows the frequencies at which INPUTS arrive, PROCESSES are performed, and OUTPUTS are produced. It may be reviewed by the user to verify that these frequencies are accurate.

Dictionary Report can show any of the following information: the person who defined the name, any KEYWORDS associated with it, a list of all SYNONYMS, and the DESCRIPTION stored in the data base for that name. A Dictionary Report showing at least names and DESCRIPTIONS can be an effective tool for communication between the systems analyst and the user. It enables the user to verify that the definitions stored in the data base for various data objects and PROCESSES correspond to the information as the user knows it. If not, he may not be effectively communicating his requirements to the analyst.

Table 7.3 Table of Contents of System Definition Report

8. PROJECT MANAGEMENT

8.1 Tools and Conventions in Using URL/URA for Project Management

This portion of the Guidebook is directed toward people who directly supervise the systems analysts who perform the analysis and description of the requirements for a system. Such a person may have overlapping duties. He may actually be a member of the overall management for the software development effort, or he may be a working systems analyst assigned to coordinate the work of the group of analysts. If he has duties in addition to those of project leader, he should read all of the relevant sections of this Guidebook.

The Project Leader is interested in

- project organization,
- keeping a record of work assignments given to the project team member,
- assuring intercommunication links between all project-team members, even those working in remote sites of the object organization,
- monitoring the overall work in progress,
- monitoring the individual performance of team members,
- detecting incompleteness and inconsistencies between the URL statements made by various analysts,
- the content and form of the final documentation generated for the external (in relation to the team members) users,
- monitoring the project progress against budget and time schedules.

In order to obtain full benefits of URL/URA assistance in project management, some standards for the use of analyst-defined names of KEYWORDS, ATTRIBUTES, ATTRIBUTE-VALUES, SOURCES and PROBLEM-DEFINERS have to be established before the team effort is begun on the project (see Section 4.1 and 4.2). Table 8.1 identifies reports available for the project leader, and their purpose. Since a single report may have multiple uses, some of the reports are listed under more than one category. Several examples of task assignment to project team members by the project leader have been given in sections 3.4.1 and 4.3.

Table 8.1 URA Reports for Project Leaders

Purpose	Reports
Complete information about specified objects	Formatted Problem Statement
Display of properties of objects	Attribute Report
Lists and dictionaries	Dictionary Report KWIC Index Name Generation Name List Punched Comment Entries
Display and analysis of data structure	Consists Comparison Report Consists Matrix Report Contents Report Identifier Information Report Picture Report
Display and analysis of data usage	Data Process Report Extended Picture Report Picture Report Process Input/ Output Report Security Analysis Report
Display of processing functions and structure	Extended Picture Report Picture Report Process Input/ Output Report Structure Report
Analysis of dynamic aspects	Frequency Report Process Chain Report Dynamic Analysis Report
Project Monitoring	Data Base Summary Projected Cost Report

8.2 Project Organization for CADSAT Use

The project organization appropriate for a particular project will depend on the size of that project, for example, the number of analysts involved and, to a lesser extent, on the type of application. In general, where several analysts are working on the same system, it is probably wise for a single person to have the responsibility for locating and correcting inconsistencies among the requirements stated by various analysts, and for finding indications that more information is needed. Depending on the size of the project, this person may be one of the group of analysts, the project leader, or an analyst assigned solely to the coordination function. The use of CADSAT should reduce the amount of direct interaction between the analysts. Rather than directing questions to each other periodically, they may simply refer to the data base for information on other portions of the system.

8.3 Monitoring the Overall Work in Progress

The Data Base Summary report gives an overall view of the quantity of information in the data base. For each object type, it presents the number of names of that type in the data base, the number and percentage of those which have synonyms, and the number and percentage which have descriptions. Totals over all object types are also presented. Comparing this week's Data Base Summary with last week's will give some idea of the volume of information which has been put into the data base in the past week. In addition, it may indicate where most of the effort has occurred. For example, if a sudden increase in defined data objects occurs, it will be clear that most of the effort has gone into data definition, while a great increase in the number of PROCESSES or in the percentage of PROCESSES having descriptions indicates a large effort put into processing definitions.

8.4 Monitoring Individual Performances

The RESPONSIBLE-PROBLEM-DEFINER statement in URL enables the project leader to monitor the work of a particular analyst. If the statement is used consistently, the NAME-GEN command can be used with the parameter:

```
SELECTION='PD=analyst-name'
```

to select the names of all URL objects which have been defined by the analyst specified. These names can then be used as input to another report, for example, a Formatted Problem Statement, showing all URL for which that analyst is responsible. Or,

```
NAME-GEN SELECTION='PD=analyst-name AND PROCESS'

PROCESS-INPUT-OUTPUT DESCRIPTION PROCEDURE
```

will clearly present the PROCESSES that the analyst has defined.