AN INTRODUCTION TO STRUCTURED SYSTEMS ANALYSIS

INFORMATICA 2/92

Keywords: System analysis, Information systems

Talib Dimij
Univerza v Ljubljani, Ekonomska fakulteta

ABSTRACT: This paper deals with well-known and very useful method for information systems development. This method is Structured Systems Analysis & Design. The reason for this work is to introduce an excellent tool to the system analyst briefly and efficiently. The method consists of two parts: Systems analysis and Systems design. This paper deals with Systems analysis, because the development of a reasonable logical model is the first step to successful development of the information system.
ABSTRAKT: Članek obravnava znano in zelo uporabljivo metodo za izgradnjo informacijskih sistemov. To je metoda strukturirane sistemske analize. Razlog, ki me je vzpodbudil k temu delu je, da uporabniku oz. sistemskemu analitiku predstavim to odlično orodje na kratek in učinkovit način. Metoda je sestavljena iz dveh delov in sicer: sistemska analiza in sistemsko načrtovanje. Ta članek obravnava sistemsko analizo, kajti razvoj zadovoljivega logičnega modela je prvi korak k uspešnemu razvoju informacijskega sistema.

Introduction

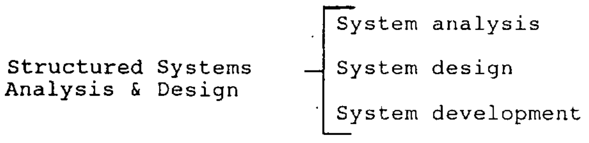
Computer-based information systems are developed in a sequence of steps. This sequence is known as system life cycle.

A huge range of detailed activities is needed to build a computer-based information system. Typical activities include writing a program, designing a form or selecting a piece of equipment. For this reason it is more common to define the life cycle as a sequence of higher level phases. Each phase is made up of more detailed activities than the last, and each has a specific goal (6).

The usual way to define the detailed activities of each phase is to review each phase when it is completed. The review produces a report on the outcome of the phase and also defines the goal as well as a detailed plan for the next phase.

Structured diagram of Structured systems analysis and design is shown in Figure 1. This paper intends to discuss the most important part of this method, this is the System Analysis.

Figure 1. Structured Systems Analysis & Design

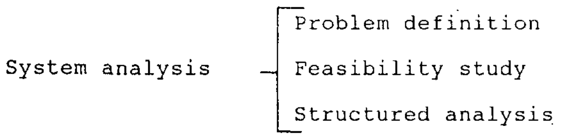


System Analysis

The key to use System analysis is the building of logical model of the required system and statment of systems objectives and constraints.

Figure 2 shows three important phases of the System analysis.

Figure 2. System analysis

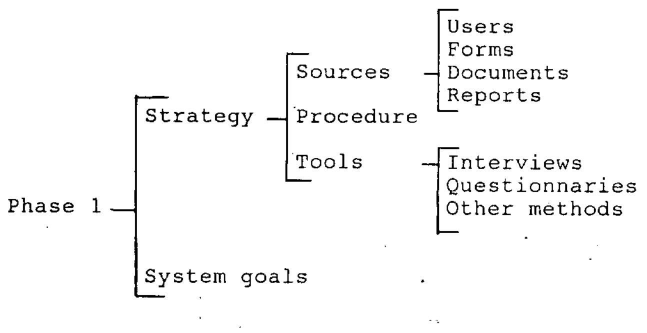


Phase 1 - Problem definition

This is a very important phase. It defines the problem to be solved and sets the direction for the whole system. It also defines the system bounds, which define what parts of the system can be changed and what parts are outside its control. Three important factors are the output of this phase. These factors are the system goals, bounds, and resource limits.

Information must be gathered in an organized way to ensure that nothing is overlooked and that all system detail is captured. All users must be consulted to ensure that every system problem, user requirement and objective is defined.

Figure 3. Problem definition



Search strategy:
It is necessary to develop a search strategy for gathering information. Search strategy is formulated by two important elements. First, identifying the information sources, and then establishing the search procedure for getting the information.

Sources: There are a number of sources of information for a system, such as system users, forms, documents, reports and so on.

System users are usually the firm information source investigated by analysts. From users it is possible to find out the existing system activities and to determine the user objectives and requirements. The usual search methods here are interviews or sometimes questionnaries.

Forms and documents are useful sources of information for system data flows and transactions. The search begins with the analyst obtaining a list of such document from the system users. Analysts then go through the documents to find their data elements and data structures.

Reports indicate the kinds of outputs needed by users. It can be used as a basis for user interviews to determine any new output requirements that user may have.

Procedure: search procedure defines where to begin the search and how to continue. It also indentifies the sequence in which sources will be searched. Search procedure is a top-down approach. That's why, it is better to begin at the top level and proceed to the bottom level to get detailed information. This approach usually commences with a set of interviews with top-level managers to determine the main system functions and activities. The analyst then searches for more detailed information to describe these functions and activities. This detailed information is obtained by interviewing operational personnel.

As soon as possible after the interview has ended, transcribe your notes. Ideally, the notes should concentrate on key ideas; use your memory to fill in the details. Share your summary with interviewee; it provides an excellent opportunity for correcting misunderstandings.

Tools: A number of search methods can be used to gather informtion about a system. These methods are interviewing techniques, questionnaries, and studying documents and reports.

Interviewing is the most common method of gathering information from users. Interviewing is a continuous process that is used by the analyst to gradually build a model of the system and to gain understanding of any system problems.

It is usually wise to begin interviewing at the top levels of the organizational areas, in order to get support and cooperation from management before beginning to look into particular organizational activities. The interview process follows a fairly structured path. We gain an appreciation of the overall system operation from management, then go into detailed operations by interviewing system users at various levels of system operation.

We should not expect to obtain all of the information required from one user in the course of one interview . There is usually a series of two, three or even more interviews with a given user. Usually the analyst begins with an initial interview to meet the users. The first interview is then followed by a number of interviews to gather all the major facts known to the user. Then there may be one or more reinterviews to verify these facts and any models developed by the analyst and to gain additional information to complete the analyst's study.

Effective interviewing is the result of careful preparation. The interview plan specifies the purpose of the interview, the users to be interviewed, the sequence in which the users are interviewed, and specific questions for every interview. A well-conducted interview consists of three distinct parts: an opening, a body, and a closing.

The Opening is the key objective to establish rapports. Begin by identifying yourself, the topics you plan to discuss, and the purpose of the interview. Tell the user why he or she was chosen for the interview. Where appropriate, identify the managers who have authorized the interview.

The Body must be created carefuly. The analyst should generally begins with a relatively broad, open question, and gradually through increasingly specific follow-up questions, focus the interview on particular points of concern. During the interview, you must listen to the answers and jot down the key points. Be selective and do not record every word.

In the Closing of the interview, when you have all information you need, thank the subject for cooperating, and offer to make your written summary available for review.

Questionnaires can be used to find out information about a system instead of interviews. Questionnaires contain all questions that a user answer to provide information sought by the analyst. The questionnaire is then send to the user and replies are analyzed by the analyst. Questionnaires are useful for gathering numerical data or getting relatively simple opinions from a number of people, but they are not very effective for in-depth searches or for identifying system problems or solutions.

Other search methods are involve studying and going through sources such as documents, reports or computer programs and looking for relevant information. Documents, reports and forms are important for identifying data elements and data structure.

System goals:
The main porpuse of the problem definition is to define the goal of the proposed system.

One important guideline for goal setting is to remember that goals should not be unrealizable ideals that are subsequently ignored. They must developed with the practicalities of the organization. One way to ensure that such practicalities can be met is to elaborate the system goal into more detailed subgoals that consider organization constraints. These subgoals are used in later stages to guide detailed analysis and design.

Another guideline for goal definition is the identification of deficiencies in the existing system. The system goal is to remove subdeficiencies. Deficiencies are usually found through interviews or by examining documents about the system performance.

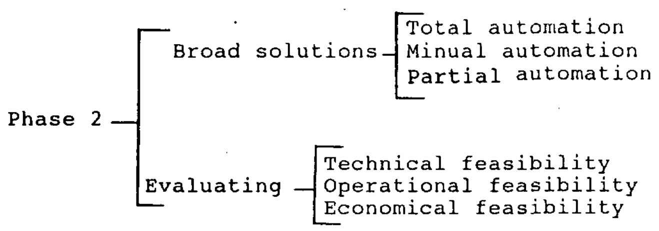
Above mentioned goals are found by examining the internal operation of a system. Goals can be examining a system external environment. It is always worthwhile to examine what similar organizations do and see whether some of their methods can be used in the proposed system.

Phase 2 - Feasibility study

Problem definition and feasibility study are important because they set the direction for the remainder of the system delvelopment.

As it was mentioned, the first phase defines the system goals. The feasibility study phase determines a feasible way for achieving this goal. It begins once the goals are defined and agreed upon.

Figure 4. Feasibility study



Broad solutions: Feasibility study starts by generating broad possible solutions, which are used to give an indication of what the new system should look like.

Two things must be kept in mind when proposing a broad solution. First it should give people a good idea of what the new system will look like and also convince them that it will work. Another objective is to form an estimate of cost. The solution should also specify what is to be done by computer and what will remain manual. So, what is needed in this phase is a broad statement of the kind of information that will be made available to users and its effect on user operations.

Usually a number of broad solutions are proposed and evaluated to find the best solution. One guideline here is to start with three alternatives, namely:

- a totally automated system;
- a system with minimal automation; and
- a system somewhere in between.

Evaluating the proposal: Three things are done in the evaluation of the proposal:

- **Technical feasibility** is the technology needed for the proposed system. If the technology is available, can it be used in the organization.

- **Operational feasibility** gives informtion about the system efficiency. Can the proposed system fit in with the current operations? Does it provide the right information for the organization's personel? Does this information arrive at the right place in time?

- **Economic feasibility** provide a reasonable estimate of system cost. Is it worthwhile to pursue the system? Many organizations place a great emphasis on economic analysis. Such economic analysis is described as cost-benefit analysis.

Cost-benefit analysis:
It usually includes two steps. One is to produce the estimates of costs and benefits. The other is to determine whether the system is worthwhile once these costs are ascertained.

The goal of first step (**producing costs and benefits**) is to produce list of what is required to implement the system and a list of the new system's benefits.

Cost-benefit analysis is always clouded by both tangible and intangible items.

Tangible items are those to which direct values can be attached. Some tangible costs are:

- Equipment costs for the new system.
- Personnel costs. These include personnel needed to develop the new system (analysts, designers and programmers).
- Material costs. These include manual production and other documentation.
- Other costs. These include consultant's costs, travel budgets and so on.

Intangible items are those whose values cannot be precisely determined. For example how much is saved by completing a system earlier or providing new information to decision makers.

The sum value of costs of items needed to implement the system is known as the cost of the system. The sum value of savings made is known as the benefit of the new system. Once we agree on the costs and benefits we can evaluate whether the project is economically viable.

Determining whether a system is worthwhile, we can use the present value method.

The idea of **the present value method** is to determine how much money it is worthwhile investing now in the order to receive a given return in some years. For example, the present value of $16,105 in Year 5 is $10,000 today at 10 per cent interest. If a system that pays back $16,105 in Year 5 costs $11,000 today, that system is not worthwhile. On the other hand, it is worthwhile if it only costs $9000 today.

To some extent the present value method works backwards. First, the system benefits are estimated for each year from today. Then, we compute the present value of these savings. If the system cost exceeds the present value then it is not worthwhile.

Let us see whether the system is worthwhile at a discount rate of 10 per cent. To do this we find the present value of the benefit at each year. The formula is:

$$\text{Present value} * (1 + r/100)^{n} = \text{Benefit at Year } n$$

$$1/(1 + r/100)^{n} \quad \text{is the discount factor}$$

Thus, for example, the present value of the $40,000 benefit at Year 2 is computed as:

$$\text{Present value} = 40{,}000 \;/\; (1 + 10/100)^{2} = 33{,}057$$

Phase 3 - Structured Analysis:

Structured Analysis is defined as the use of Data flow diagrams, Data dictionary, Structured English, Decision tables, and Decision trees, to build a target document, the structured specification (4).

These tools provide a communications mechanism between the analysts and the users, and a precise means of recording the specification so that the design can be translated into computer code and implemented (2).

Figure 5. Structured Analysis

- Structured Analysis
  - Data flow diagram
    - Processes
    - Data flows
    - Data stores
    - Data sources
  - Data dictionary
  - Structured English
  - Decision tables & trees

Data flow diagram:

The data flow diagram (DFD) is one of the most important tools used by analysts. DFD is the logical model of the system. The model (DFD) does not depend on hardware, software, data structure, or file organization.

A data flow diagram (DFD) is a charting tool which traces a network of data flows through a system and provides information at varying levels of detail. This enables the system requirement to be partitioned, analyzed and specified in manageable pieces (2).

DFD uses four kinds of symbols. These symbols are used to represent four kinds of system components: process, data store, data flow and data source (external entity).

Process: Processes show what system do. Each process has one or more data inputs and produces one or more data outputs. A process transforms incoming data flows into outgoing data flows. Process has a unique name and a number. This name and number appear inside the circle that represents the proces in a DFD. A single process is not necessarily a program. It might represent a series of programs, a single program, or a module in a program.

Data store: A data store is a repository of data. It might represent a data base, a file, or even a piece of file. Each data stores are represented by a thin line, closed at one end. Each data store can be identified by a "D" and an arbitrary number and a name inside the box.

Data flow: Data flows model the passage of data in the system and are represented by lines joining system components. The direction of the flow is indicated by an arrow and the line is labeled by the name of the data flow.

What is the difference between a data store and data flow? A data flow is data in motion; a data store is data at rest (3).

External entity: External entity is an individual or organizational unit outside the boundary of the system that interfaces with system. External entities are either supply input data into the system or use the system output. External entities that supply data into a system are also called sources, and external entities that use system data are sometimes called sinks. External entities are represented by a rectangle.

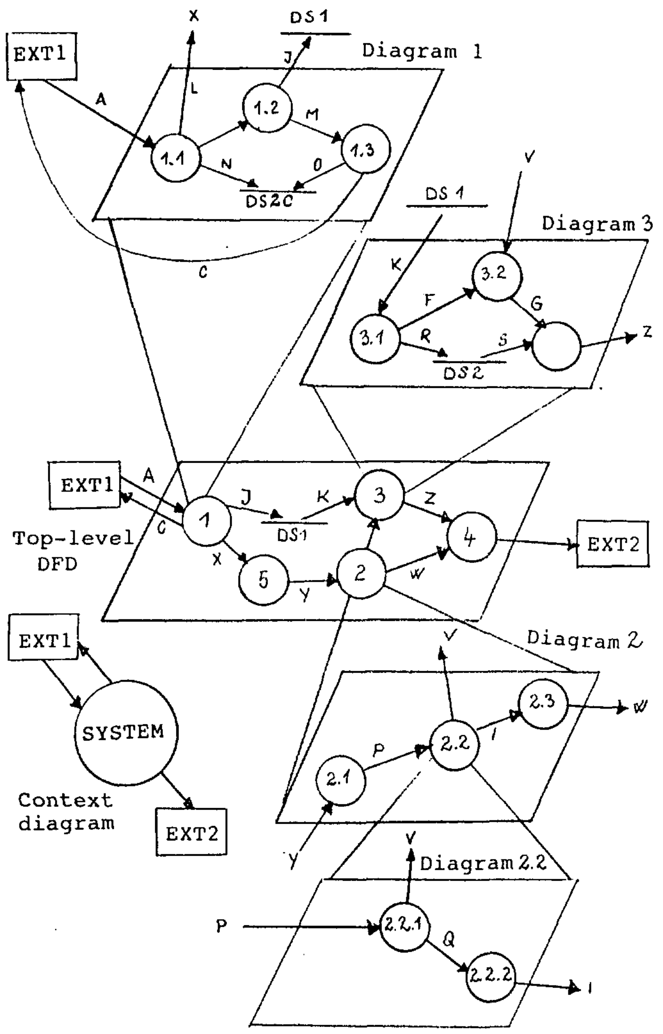
Leveling data flow diagrams: The drawing of data flows is a highly interative process. The best way to begin is to model the whole system by one process. This process is the highest level data flow diagram. It is called as contex diagram.

Context diagram shows all the external entities that interact with the system and the data flows between these external entities and the system.

Each process can be exploded to a further level of detail until the lowest level, called a functional primitive elementary process is reached.

A primitive is a process which generally executes a single function and can be described in less than a page of structured English (2). When the lowest level consists only of primitive, the leveling is complete.

Figure 6. Leveling



Leveling allows an analyst to start with a top-level function and elaborate it in terms of its more detail components. Although leveling is very useful tool, a number of conventions must be observed to ensure that no information is lost as a DFD is leveled. Some these conventions are:

a) Process numbering: The context diagram is usually given the number 0. Processes in a next level (top-level DFD) are numbered consecutively, starting with 1 and continuing until all processes have beeh labeled. This level shows the major processes in the system. In Figure 6 the top-level DFD has five processes numbered 1, 2, 3, 4 and 5.

As each process is leveled, its DFD diagram is given the same number as the process. Thus, in Figure 6, the leveled DFD of process 1 is named Diagram 1. Each process in the leveled DFD receives a number made up of its diagram number, follow by a period, followed by a number within the leveled DFD. Thus the processes in DFD Diagram 1 have process number 1.1., 1.2. and 1.3.

b) Data flow balancing: Data flow balancing requires that all the data flows entering a process are same as those entering its leveled DFD. Similarly, all the data flows leaving the process are the same as those leaving its leveled DFD. If you look at Diagram 2, which is the leveled DFD of Process 2 in the top-level DFD. You should note that the inputs (Y) and the outputs (V and W) of Process 2 are the same as the inputs and outputs of Diagram 2.

c) Data stores introducing: Data stores are local to the leveled diagram. In Figure 6 data store DS2 contains data that is local to Process 3. Hence this data store does not appear in the top-level DFD but only in its leveled diagrams.

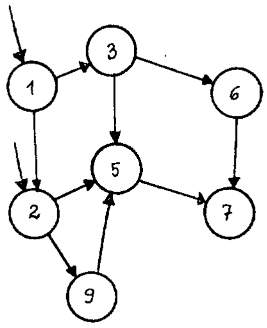
d) External entities introducing: All external entities should appear on the context diagram. They should also appear with a DFD if any process in the DFD has a flow to or from the external entity. In Figure 6, external entity EXT1 has flow to and from Process 1. This external entity appears on the leveled DFD for Process 1.

It is sometimes difficult to know far to level down, how many processes to include on a data flow diagram or even where to start. It is imposible to give precise answers but some guidelines can be suggested.

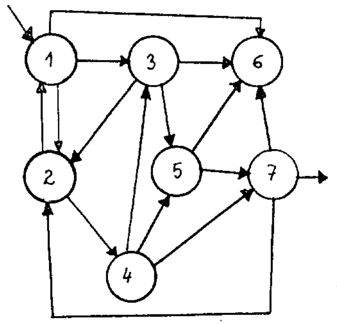
Most practitioners say that about seven plus or minus two is the ideal number of processes on a data flow diagram. This number can be clearly understood by a visual examination but is not too small to be trivial. A larger number is sometimes too hard to understand; smaller number often includes too little information to be useful.

Figure 7. Reducing complexity by minimizing flow between processes

a) Minimized data flow



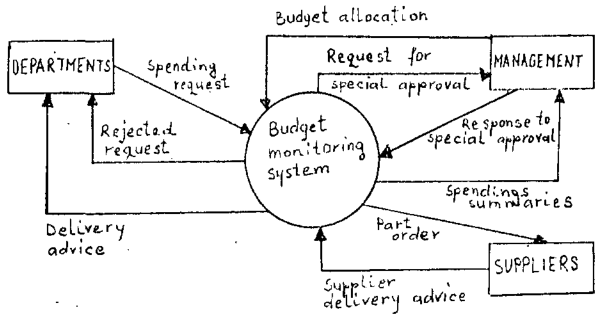
b) Too many flows



However, one should not take a trivial approach and suggest that leveling only involves taking a higher-level process and partitioning it into seven lower level-processes. Consider Figure 7. It contains two data flow diagrams, each with seven processes. One of these diagrams is clearly less complex and hence easier to understand than the other. Figure 7 illustrates another requirement of leveling - the interfaces between processes must be minimized. The analyst must carefully analyze the top-level process. If the interconnections are too complex, the leveling should be re-examined and restarted. This may continue over a number of iterations until a satisfactory solution is found.

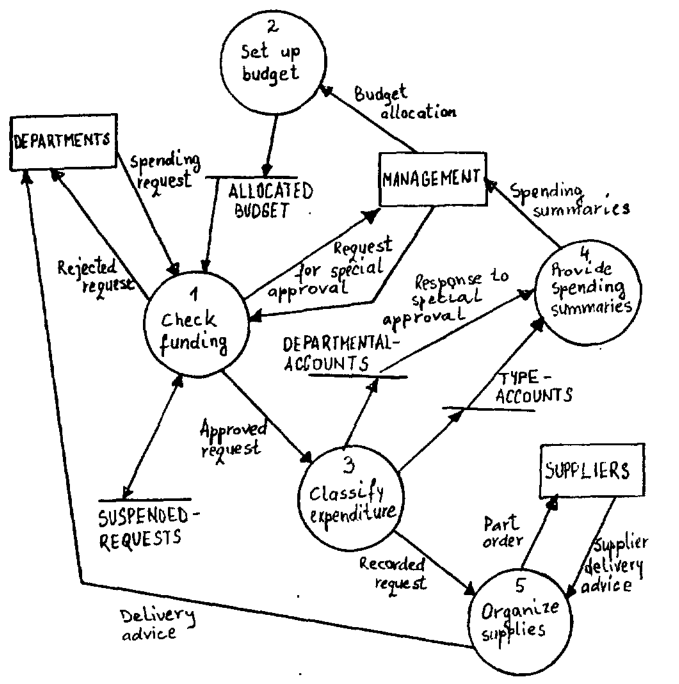
Example: Figure 8 is an example of a context diagram. It models the "budget monitoring system". This system interacts with three external entities, DEPARTMENTS, MANAGEMENT and SUPPLIER. The main data flows from DEPARTMENTS are "spending request". In response, DEPARTMENTS either receive "rejected request" data flows or "delivery advice" data flows. MANAGEMENT receives "request for special approval" data flows to which it responds. MANAGEMENT also sends "budget allocation" data flows to the system and gets "spending summaries" data flows. Suppliers receive "part order" data flows and return "supplier delivery advice" data flows.

Figure 8. Context diagram



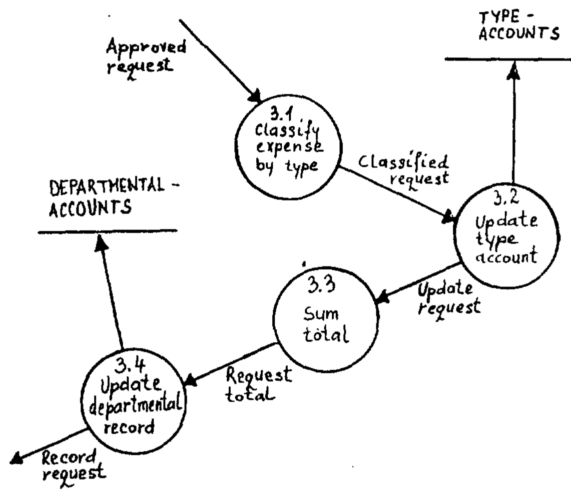
This model does not describe the system in detail. For more detail it is necessary to identify the major system processes and draw a data flow diagram made up of these processes and the data flows between them. The DFD that shows the major system processes is called the top-level DFD (shown in Figure 9). The top-level DFD shows the various processes that make up the system. Each process has a unique name and number. From Figure 9 we see that data flow "spending request" from DEPARTMENTS goes to the "check funding" process. This process looks up the allocated budget and determines whether special permission is needed from MANAGEMENT to proceed with the request. Approved requests go to the "classify expenditure" process where they are entered into data stores, DEPARMENTAL-ACCOUNTS and TYPE-ACCOUNTS. Finally, if required a "part order" for parts originally specified in a "spending request" is placed with suppliers. There are two other processes. One is to "set up budget" and the other to "provide spending summaries".

Figure 9. A top-level DFD diagram



We can continue to expand each process in Figure 9 into a more detailed FDF. As an example, Figure 10 is a detailed description of the "classify expenditure" process labeled as Process 3 in Figure 9. Each process in this detailed DFD is labeled with a 3 followed by a number to show that each is an expansion of Process 3. The input to this DFD is the data flow "approved request". Process 3.1 classifies the entries in each "approved request" by expense type and Process 3.2 uses this classification to update data store TYPE-ACCOUNTS. All the entries for the whole request are summed by Process 3.3, and Process 3.4 uses the total to update data store DEPARTMENTAL-ACCOUNTS.

Figure 10. Expansion of "classify expenditure" process



Data dictionary:

The purpose served by the data dictionary during structured analysis is to maintain definitions of data flows, data stores, and processes. The basic information included in each data flow or data store entry is its name, aliases and composition.

The data flow name identities the content of the data flow and the action taken on it. The alias is any other name by which the data flow may be known.

The compositon can be defined using the convention notations described by DeMarco. The conventions are:

= means is equivalent to, + means and,[ ] means either-or,{} means iteration of the components enclosed, and ( ) means that the enclosed component is optional.

For example the ORDER document can be described as

$$\text{ORDER} = \begin{bmatrix}\text{SUPPLIER-NO}\\ \text{SUPPLIER-NAME}\end{bmatrix} + \text{DATE-ORDERED}$$

$$+ \text{DATE-REQUIRED} + \text{ORDER-NO} + (\text{ORDER-STATUS}) + \{\text{ITEM-NO} + \text{QTY-ORDERED}\}$$

Here ORDER is the structure name. The order is made up of seven data elements: SUPPLIER-NO, DATE-ORDER, DATE-REQUIRED, ORDER-NO, ORDER-STATUS, ITEM-NO and QTY-ORDERED.

In this notation, structure components are described by using the plus sign (+). The square brackets [ ] indicate alternative structures. Thus ORDER may contain SUPPLIER-NO or SUPPLIER-NAME but not both. The round brackets ( ) indicate an optional component. Thus an ORDER may or may not contain the data element ORDER-STATUS. The curly bracket { } indicate that a structure or data elemente can be repeated. So, the structure
{ ITEM-NO + QTY-ORDERED } can be repeated.

Structured English:

Structurd English is used to give unambiguous process description. There are several variations of structure English. It is a very limited subset of the English language. Structured English syntax provides the keywords to structure process specification logic. Process specification logic consists of a combination of sequences of one or more imperative sentences with decision and repetition constructs.

Imperative sentences: An imperative sentence usually consists of an imperative verb followed by the contents of one or more data stores on which the verb operates. In imperative statements we can use commands such as move, get, write, read, compute and so on.

Data flow names appear in lower case between quotes while specific data items in the data flows and data stores are capitalized.

A sequence of imperative statements can be grouped by enclosing with BEGIN and END.

```
BEGIN
   Receive "sale report"
   Get SALES record for PART-NO in
     "sale report".
   TOTAL-QTY = TOTAL-QTY + QTY-SOLD.
   SALE-VALUE = QTY-SOLD * UNIT PRICE.
   TOTAL-VALUE = TOTAL-VALUE + SALE-VALUE.
   Write SALES record.
   Send "summary advice".
END
```

Decisions: Two types of decisions structure usually appear in Structured English. First type shows a structure which allows a choice between two groups of imperative statments. The keywords IF, THEN and ELSE are used in this structures. If a condition is "true" then GROUP A sentences are executed. If a condition is "false" then GROUP B sentences are executed.

```
IF condition THEN
   BEGIN
      GROUP A
   END
ELSE
   BEGIN
      GROUP B
   END
```

The second type shows a choice between any number of groups of imerative sentences. The keywords CASE and OF are used in this structure. The value of a variable is first computed. The group of sentences excuted depend on that value.

```
CASE NAME OF
    A:
     BEGIN
        GROUP A SENTENCES
     END
    B:
     BEGIN
        GROUP B SENTENCES
     END
    .
    .
    .
    Z:
     BEGIN
        GROUP Y SENTENCES
     END
  END
```

Repetition: For specifying itertions we use two commands. First way is to use the WHILE...DO structure. Here a condition is tested before a set of sentences is processsed. The second way uses REPEAT...UNTIL structure. Here the group of sentences is excuted first and then the condition tested.

```
WHILE  condition  DO
  BEGIN
     GROUP A sentences
  END

REPEAT
  BEGIN
     GROUP A sentences
  END
UNTIL  condition
```

An alternative to the WHILE...DO structure is to use the FOR structure.

```
FOR INDEX = INITIAL TO LIMIT
   BEGIN
      GROUP A sentences
   END
```

Decision Tables and Decision Trees:

Algorithms involving multiple nested decisions are difficult to describe using structured English, pseudocode or other techniques.

A better way of describing such logic is to use decision tables or decision trees. These two efficient methods are prefered where one of a large number of actions is to be selected. The action selected depends on a large number of conditions.

Decision tables: A decision table is first devided into two parts - the conditions and the actions. The conditions part states all the conditions that are applied to the data. The actions are the various actions that can be taken depending on the conditions. The table is constructed by using columns so that each column, called role, corresponds to one combination of conditions.

A sample decision table is shown as Figure 11. Assume that the basketball coach has asked us to look through the student records and produce a list of all full-time male students who are at least 185 cm tall and who weigh at least 85 kg.

Figure 11. Decision table

CONDITIONS					
Is the student male?	Y	N			
Is the student taking at least 12 credit hours?	Y		N		
Is the student at least 185 cm tall?	Y			N	
Does the student weigh at least 85 kg?	Y				N
ACTIONS					
List the student's name and address	X				
Reject the student		X	X	X	X

The easiest way to understand a decision table is to read the first question: Is the student male? There are two possible answers: yes (Y) or no (N). If the answer is yes; we can not make a decision. Three more tests must be passed. What if the answer is no? The student is not male, she is not a candidate for the basketball team, and thus can be rejected. Move down the column containing the N, and note the X on the action entry line following "Reject the student".
Move on to the second question: Is the student taking at least 12 credit hours? There are two possible answers: yes or no. If the answer is yes; again, we can not make a decision; two more test remain (must be passed). What if the answer is no? The student can be rejected. Move down in third column and note the X on the action entry line following "Reject the student".

Read the rest of the table. It clearly shows that the student's name and address will be listed only if the answers to all four questions are yes, but that the student will be rejected if the answer to any one is no.

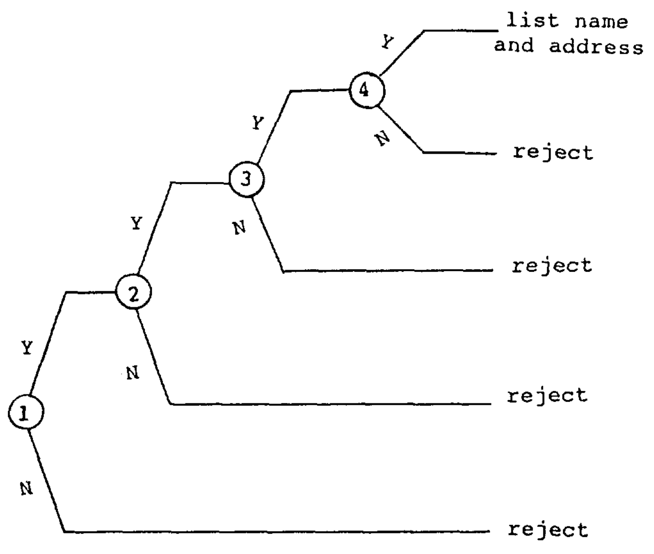
When an algorithm involves more than two or three nested decisions, a decision table gives a clear and concise picture of the logic (3).

Decision trees: As an alternative, the anlyst might use a decision science tool called decision tree. Decesion tree is easy to constuct, and graphically illustrate decision logic.

The decision tree is a graphic representation of the decisions, events, and the consequences associated with a problem. Once the tree is drawn, probabilities can be associated with each branch, and expected values of the outcomes computed (3).

Figure 12 is shown a decision tree for basketball problem logic. The algorithm consists of a series of four nested questions or decisions. Each question or decision is represented as a circle. Begin with the first one: Is the student male? There are two possible answers: yes or no. If the answer is no: the student is rejected; a yes answer leads to another question. Again, there are two possible answers: yes or no. Again, a no means reject the student, while a yes response leads to another question and so on. Follow each branch on the tree to its logical outcome. Student's name and address are listed only if all four questions are answered affirmatively.

Figure 12. Decision tree



1. Is the student male?
2. Is the student taking at least 12 hours?
3. Is the student at least 185 cm tall?
4. Does the student weigh at least 85 kg?

Conclusion:

Structured Systems Analysis as an excellent hierarchical design approach provides a well-organized and manageable system. Throughout the process, we deal with manageable pieces of information which can be controlled. It is the step-by-step evolution of the analysis. We begin with the present physical system, convert it into the present logical system. This logical system provides us with a framework to specify a new physical system.

Litrature:

1) J. Daniel Couger, Mel A. Colter, Robert W. Knapp: Advanced System Development/Feasibility Techniques. John Wiley & Sons, New York, 1982.

2) Denis Connor: Information System Specification and Road Map. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1985.

3) William S. Davis: Systems Analysis and Design. Addison-Wesely Publishing Company, Massachusetts, 1984.

4) Tom DeMarco: Structured Analysis and System Specification. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1979.

5) Chris Gane and Trish Sarson: Structured Systems Analysis, Tools and Techniques. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1979.

6) I. T. Hawryszkiewycz: Introduction to Systems Analysis and Design. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1988.

7) Robert E. Leslie: System Analysis and Design, Method and Invention. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1986.

8) James C. Wetherbe: Systems Analysis and Design. West Publishing Company, New York, 1988.