UNIT 6

DESIGNING ON-LINE AND DISTRIBUTED ENVIRONMENTS- DESIGN CONSIDERATION

Objectives

After going through this unit, you should be able to:

l understand system design features related to on-line systems

l know about the various computer system concepts related to on-line design.

Structure

6.1 INTRODUCTION TO ON-LINE OR REAL - TIME ENVIRONMENT

Data processing on computers, in its conventional form, consisted mainly of batch-processing - sequential or indexed-sequential, much after the event for which the processing was taking place. However, with sophisticated need for computers, the need for on-line or real-time processing came up. On-line or real-time processing is the processing of data on computers to produce the requisite results in very short time to enable an event to take place. Some of the activities which take place in real-time processing with computers are:

i) **Computers - aided Manufacturing :** Where processes like thermal power generation are controlled by computers

ii) **Computers and Space :** Where the launching and journey of rockets/satellites in space is controlled by computers

iii) **Computers in Medicine :** Where vital activities of the body, like electrocardiogram, are controlled by computers

Activity A

From your experience give five more examples of real-time systems other than those mentioned above.

.....

.....

.....

.....

6.2 ON-LINE SYSTEM ANALYSIS AND DESIGN CONSIDERATIONS

In an earlier unit you have been introduced to system analysis and design. Let us see here what are the factors that influence on-line system analysis and design.

Before we take up the above mentioned task, let us define the term **system**. A system is a group of interrelated

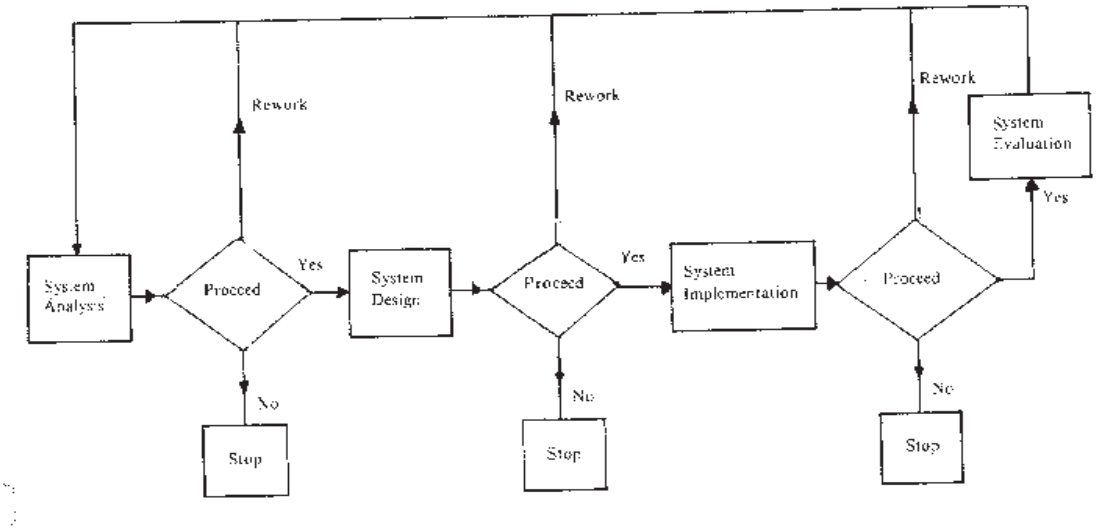

Figure 6.1 : System Development Life-Cycle

subsystems interacting with each other and the environment to achieve desired objective.

The following figure shows the steps involved in system development, be it on-line or off-line.

Making even minor changes without considering what has been done previously can cause errors or have unanticipated effects elsewhere.

While designing a system it should also be remembered that the final decision making regarding the end output of the system is the user's prerogative. The impetus or need for system development should come from the user community. Treating a computer as a solution-in-search-of-a-problem, instead of a tool to solve an already-identified problem can only lead to mistakes.

Even on-line system design consists of **preliminary investigation** and **feasibility study.** Detailed investigation consisting of:

- fact-finding
- data analysis and evaluation
- estimating costs and benefits, and
- preparation of a system proposal

follows.

By maintaining objectivity in fact-finding, the analyst can separate habitual or traditional acts from acts resulting from actual requirements and thus ultimately evolve a suitable system design in the data analysis and evaluation phase.

Data analysis and evaluation consists of recording, organising and evaluating collected facts. Information, system personnel begin to extract significant factors that help them identify the user's system requirements from this wealth of data. Increasingly, analysts are using computer-assisted software engineering (CASE) tools to organise and analyse data. The term CASE has been coined to refer to the automation of tools. methods and procedures used in system development. The products that accomplish the automation are computer programs, referred to collectively as CASE tools, They are grouped within two broad categories:

- **Front-end CASE tools** assist in the analysis of user requirements, description of data elements, description of operations (e.g., page sizes and numbers of copies for report printing). a high-level system design (output, input and file formats, high-level internal program structures and so on).
- **Back-end CASE tools** used to convert specifications into machine - executable codes, assist in debugging and testing, and/or restructuring existing databases and programs.

Some analysts organise their findings on flow diagrams (DFDs)

Once a system proposal has been approved by the user, a team of analysts and programmers set about coding and developing programs for the system.

Designers can use any of the several CASE tools to create mock-ups, called prototypes, for user review as early as possible.

Various methods used to design the system can be summarised as:

i) **Top-down design** : When doing top-down design we look first at the major function to be accomplished by the system and then fill in the details.

ii) Tree-diagrams or structure charts.

iii) Decision tables, etc.

Although, the techniques described above are similar to those that might be used for off-line systems, the major points to be kept in mind while designing on-line or real-time systems are as follows:

i) response time of the system should be very efficient to cater to the fast response times required in real-time systems;

ii) scope of failure of the system due to power failure or bugs in the system should be minimised, otherwise it may lead to fatal system break-downs;

iii) a certain degree of self-correction and review methods should be inbuilt in the system; and

iv) on-line system design should take into consideration the capabilities of the computer system- like the number of tasks/ programs it can handle at a time, number of terminals which can be attached to it, etc.

After the system analysis and design phase, we come to the system implementation phase. However, it must be borne in mind that no amount of good coding can repair the damage done by poor design or program coding. While there is no universally accepted definition of what constitutes a high quality system/program, most users agree that the following characteristics are important:

i) **Correctness:** Extent to which a program/system satisfies its specifications and meets the user's needs;

ii) **Reliability :** Extent to which a program/system performs its intended function without error;

iii) **Robustness :** Extent to which programs/system accommodates and handles unacceptable input and/or user operational errors. This factor is especially important for on-line systems;

iv) **Usability :** Effort required to operate, prepare input for, and understand output of a program;

v) **Testability:** Effort required to test a program to insure it performs its intended function;

vii) **Maintainability :** Effort required to locate and fix an error in a program/system. This is also an important factor in a program/system operating in an on-line environment;

vii) **Extensibility :** Effort required to modify a program/system to change or enhance its function. Again, an important factor in on-line systems.

viii) **Efficiency :** Amount of computing resources required to use a program/system.This is also an important factor in real-time system design since space required for data and programs should be large enough to cater to the system requirement;

ix) **Portability :** Effort required to transfer a program from one hardware and/or software environment to another one;

x) **Reusability :** Extent to which a program/system can be used in other related applications; and

xii) **Turn- around time :** Time required from feeding of input to receiving the output. This is important in on-line systems.

Activity B

What are the design considerations in on-line systems which are especially important for it or different from conventional systems and in what way? Give examples.

.....

.....

.....

.....

The following methods can be adopted for system implementation:

a) Cut-over to new system and leaving the old system abruptly. This suffers from the drawback that if the old system is better or the new computerised system has drawbacks, then the new system implementation becomes very difficult. Bringing back the old manual/semi-manual system is also difficult. Hence this system of implementations not advised for on-line systems.

b) Phasing out the old system and bringing in the new computerised system in stages. This is better than the cut-over system, yet in most cases it is difficult to implement in on-line systems. In some cases, like railway or airline reservations, it may be useful.

c) Parallel run of the old system and the new computerised method till such time as the new system has established. This is better than the first two methods for on-line systems, as it gives time to new on-line system to stabilise.

d) Installation of a new on-line system. This requires repeated and meticulous testing for on-line systems before it can be adopted. This is true in cases like rockets/satellites launching and control.

6.3 SELECTING A LANGUAGE

How does one choose from among the myriad of programming languages the language that is most apporpriate for a particular system development effort?

Let us back up a bit. The first phase of the system development cycle is system analysis. The second phase is

system design. In most cases, after the user requirements are understood, alternantive solutions can be identified. Today, most if not all user organisations are confronted by resource constraints. The programming talent, time and money needed for in-house system development are likely to be in short supply. Therefore , if application software packages that meet or nearly meet user requirements are available, they should be considered. Perhaps a “nearly-meets-requirements” package can be customized to do the job.There is no merit in writing a new program if an acceptable one already exists. However, if detailed program design, coding and checkout are to be done, then the selection and use of one or more programming languages is required.

Usually, one assembler program, or a system software program written in a language like “C”, is provided for each particular family of computers. Programs can be written for the computers in the assembler language that the assembler program is designed to translate. Theoretically, any number of high-level-language processors can be written for any computer. It follows that any computer can be controlled by programs initially coded in any of several high-level programming languages, depending on the language processors available.

In deciding which programming language to use, some specific questions should be asked. Which programming languages are available? What knowledge does the head of the programming from assigned programmers or user- programmers have of those languages? What types of problems are the various languages designed for? These and other language selection considerations are summarized in Figure 6.2 below:

- LANGUAGE AVAILABILITY
- LANGUAGE KNOWLEDGE OF SYSTEM TEAM
- SUITABILITY OF THE LANGUAGE TO THE PARTICULAR PROBLEM OR APPLICATION
- EASE OF LEARNING THE LANGUAGE
- UNDERSTANDABILITY AND DOCUMENTATION EASE
- EASE OF MAINTENANCE
- LANGUAGE STANDARDIZATION AND PROGRAM PORTABILITY
- SPEED OF SOURCE-PROGRAM TRANSLATION BY THE LANGUAGE PROCESSOR AND LANGUAGE-PROCESSOR STORAGE REQUIREMENTS
- EFFICIENCY OF RESULTANT OBJECT PROGRAMS IN TERMS OF NUMBER OF INSTRUCTIONS, EXECUTION SPEED AND STORAGE REQUIREMENTS. THIS IS ESPECIALLY IMPORTANT FOR ON-LINE APPLICATIONS

Figure 6.2 : Factors to be considered for chossing a language that is most approperiate for a system development effort.

The relative importance of the selection criteria varies depending on the situation.

As we have seen, FORTRAN is well suited for problems that involve mathematical computations. COBOL was designed with business applications in mind. BASIC was designed to provide immediate, straight forward answers to simple problems. For many end-user real-time applications, a fourth generation language may be the appropriate choice.

The assembler language of a computer can be used to express all the operations the computer is capable of performing. If fast execution and/or minimal use of storage by the resultant object program are of primary importance, use of assembler language may be advisable. Some fourth generation languages’ tools allow assembler-language routines to be embedded in fourth generation language applications to perform time-critical functions. This is especially important for on-line systems.

6.4 APPLICATION AND SYSTEM SOFTWARE REQUIRED FOR ON-LINE SYSTEMS

As you may be aware, software is of two types (i) application software, and (ii) system software.

To recall, system software is that part of the software that performs the tasks of controlling the input, output CPU functions and other instruction-related portions that are performed by the hardware of the computer system. Application software are the programs written by programmers to achieve the end-objective of the user community of the computers. System software acts as a 'middleman' between the application software and hardware of the system.

The conventional batch-processing techniques worked well where large volumes of input data were required for processing with significant time interval between one lot of input data and its required output and the next lot of input data sent for processing. For example, the processing of weekly time cards for all employees. They did not work well for small amounts of input generated randomly or for individual problem solving. Computer manufacturers addressed these needs by developing on-line direct-access systems.

In an online direct-access application, the computer communicates directly with both the source and the destination of the data it processes. The data can be sent to and received from local I/O devices or to and from I/O devices at remote locations by way of communication channels. Input transactions can be processed as they are received. Master files can be read to produce up-to-date output information in the form of status reports, invoices and so on. This approach is called **transaction processing**. Because queries about master-file data are received in random order, the files are generally stored on direct- access storage devices.

The first on-line direct-access systems were typified by the early airline reservations system.

These early on-line direct-access system met many user needs, but they were far from perfect. Because the systems had to respond quickly to many individual transactions, it was neither possible nor desirable to query or update master files. Instead, the master files had to be updated continuously. Otherwise, for example, a ticket agent at one location might sell airline space already sold by an agent at another location.

Further more, different types of transactions, say inquiries, sales and cancellations, required different programs to process them. These programs had to be kept in secondary storage and brought into primary storage when needed. Much time was spent in locating and gaining access to data and programs, rather than in processing transactions.

To add to the problem, transactions occurred irregularly. During peak periods, hundreds of thousand of transactions had to be processed within minutes. At other times, the systems were relatively idle. It become apparent that to use EDP system resources as efficiently and effectively as possible, new techniques had to be developed.

6.5 MULTITASKING OR MULTIPROGRAMMING

Fortunately, during this same time period, system software and hardware developments were occurring elsewhere. In 1963, the Burroughs Corporation released its Master Control Program, or MCP, for use with Burroughs B 5000 computers systems. MCP assumed greater control over system resources than its predecessors had. Input and output devices were activated by MCP rather than by application programs. This centralised control was possible because hardware interrupt conditions signalled to MCP when control of the EDP system should be passed to a special-purpose routine for I/O processing.

MCP could also assign memory areas to programs, determine the optimum sequence and mix of jobs, determine program priorities and system requirements, and provide for rescheduling if new jobs were introduced or job priorities changed. More than one user application program could be resident in primary storage at a time. Since the processor could execute instructions from different programs was not possible. The processor could, however, execute only one instruction at a time, simultaneous execution of instructions from the first program, then instructions from another program, than instructions from one program again, and so on. This type of processing is called **concurrent processing.** Program logic within MCP determined which system software routine or application program had control of the B 5000 prcessor at any given time. Today, a system that provides this capabilities is said to support **multitasking** or **multiprogramming.**

Concurrent execution of programs is desirable because I/O operations are much slow than internal processing operations. Fortunately, the design of modern computer systems permits overlapping of I/O and processing operations.

Under one approach, the actions of I/O devices are permitted to occur only at fixed points in a program and only in a sequence established by the program. Such a system is said to support **synchronous operations.** As an example, assume a user wants to key in data from a VDU. Let's suppose a particular bit in storage is set to 1. The setting of the bit does not cause a read operation to occurimmediately, however. At fixed intervals , the processor is directed to rest that bit position. It also tests the bit positions reserved for other I/0 devices on the system. This technique is called **polling**. When the VDU's turn comes and its bit is found to be set, the input operation is allowed to occur. Since internal processing operations occur very rapidly, the user may think the data keyed in is read immediately. That is not the case. All activities within the system are rigidly controlled by the hardware and/or by stored programs.

Other EDP systems perform **asynchronous operations**. Such systems are designed to permit the automatic interruption of processing whenever the need for I/O activity arises. The input or output device signals the processor by means of an **interrupt** when it is ready to read or to write. The interrupt means, in effect : "My particular job is done. As soon as convenient, use the data I have given you (if it was an input operation) or give me any additional information you have (if it was an output operation). The signal may be the setting of a bit as above, but in this case, the bit has an immediate effect. As soon as the processor finishes whatever instruction it is executing, it accepts the data as input or transmits the information as output.

Ideally, the configuration and speeds of the various I/O devices included in a computer should be such that the processor can work at full capacity whenever the user workload dictates. Again, because of the slowr speeds of I/O devices, this means that the over-all system efficiency depends heavily on the extent to which input, internal processing and output operations can be overlapped, or allowed to occur at the same time.

Even so, if only one program is resident in primary storage and executing, chances are the processor is idle much of the time.

In 1964, IBM introduced its major third-generation operating system. OS/360, for use with IBM System/360 computers. A System/360 user could select the version of OS/360 that included a control program called **Multiprogramming with a Fixed Number of Tasks (MFT)**. Each task was simply and independent unit of work, such as a program or subroutine, that needed system resources. As its name implies, OS/MFT could operate on a fixed number of tasks concurrently. Actually, it could read jobs from as many as three job input streams, handle upto 15 job steps, and record upto 36 streams of output concurrently.

A system/360 user with extensive data-processing requirements could select another alternative: a version of OS/360 that included a control program called **Multiprogramming with a Variable Number of Tasks (MVT).** As its name implies, OS/MVT could control a variable number of tasks concurrently. It could change the number, size and location of reserved storage areas to meet the data-processing requirements at any given time. Like OS/MFT. OS/MVT could handle as many as 15 job steps concurrently. Moreover, once a job step was initiated by OS/MVT, that job step could, in turn, initiate the processing of other tasks. There was not a 1-to-1 relationship between job steps and tasks (as existed under OS/MFT).

OS/MVT was used on the very largest mainframes of the late 1960s. These big mainframes sometimes had as much as 1 mega-byte of primary storage. A big application program was one that needed 256 kilo-bytes.

The Burroughs B 1700, announced in 1972, was the first small computer with an operating system that supported multitasking. Today, there are micro-computer systems with equivalent or greater capabilities. MS-DOS is a single-user, single-tasking operating system. OS/2 is single-user, multitasking operating system. It can interact with only one user, but that user may initiate the concurrent execution of two or more programs. UNIX is a

multiuser, multitasking operating system. It can communicate at what appears to be the same time with two or more interactive users and can perform concurrent proccessing.

For real-time systems, multiuser, multitasking operating systems would be ideal.

Virtual storage capabilities allow a system to be used as though more primary storage exists than is actually present. A greater number of programs can be running at a time than would be possible otherwise.

Activity C

UNIX has been mentioned as one of the multiuser, multitasking operating systems. Can you name three more multiuser, multitasking operating systems.

.....

.....

.....

.....

6.6 MULTIPROCESSING

Let us pause a moment to clarify some concepts and to relate some things we have learned. Like multi-programming and multitasking, the terms multi-programming and multiprocessing are sometimes used interchangeably, but they do not mean the same thing. Multitasking involves concurrent execution of instructions from two or more programs sharing one processor and controlled by one major operating-system control program. In multiprocessing, instructions are executed **simultaneously** (at the same time, in parallel) on two or more processors; the processors can execute different instructions from the same program or from different programs at a given time.

In today's most basic multiprocessing systems, one main processor handles all major processing functions. Other co-processors handle "housekeeping" chores such as opening and closing files, input validation and editing, and file maintenance or perform complex mathematical functions. The more sophisticated multiprocessing systems involve several main processors. There is no single "computer", but rather a **"computer system"** that consists of several processors linked together for purposes of communication and co-operation during processing.

Multiprocessing is not limited to mainframe environments. For example, the IBM PS/2 Model 70 computers introducted in late 1988 have a 32-bit Intel 80386 main processor and, optionally, an Intel 80387 math co-processor. Apples Macintosh SE/30, announced in January 1989, have a 32-bit Motorola 68030 main processor and a Motorola 68882 floating-point (math) co-processor.

Many of the problems now being addresed with computer help involve lots of data and lots of very fast computation and feedback-for example, monitoring and/or controlling the many variables that must be taken into account when journeys into space are initiated. With multiple processors, parallel work on different problems or on the same problem is possible.

Activity D

Differentiate between multitasking, multiprogramming and multiprocessing and try to give examples.

.....

.....

.....

.....

6.7 REAL-TIME SYSTEMS

A system that can also provide output fast enough to satisfy any user requirement can be further classified as a **real-time system.** Such a system makes output available quickly enough to control (not simply react to) real-life activity.

The concept of real-time is closely related to immediacy. It is seen as **response time,** which we now define somewhat formally to be the interval of time between completion of input and start of output from an EDP system.

In actual operation, real-time is a matter of degree, depending upon the application at hand. The customer service representative of an insurance firm may be willing to wait from 3 to 10 seconds for details about the coverage of a policyholder. In a military defense system, responses within microseconds may be required. Variations in response time are due to differences in the system work-load, internal processing requirements, frequency and type of access to computer files and/or databases, and so on. Both the hardware and the software must be capable of fast performance. In addition, the system must be tunded by support personnel to fit the requirements imposed on it. For example, system characteristics such as the number and size of I/O areas, or **buffers**, may be set at system startup time.

Although basic business applications such as order writing, inventory control and payroll can be implemented as real-time system, they are not likely to be. Increasing the costs of software, hardware development and ongoing support is more likely to be justified for specialised applications in industries such as transportation, manufacturing, banking and distribution. Computer controlled robotic systems on assembly lines, automated teller machines at banks and point-of-sale terminals at retail department stores abroad are components of real-time systems. In hospitals, patient's vital signs are monitored at bedside by real-time system.

Time sharing is a technique that allows several users of an on-line real-time system to use that system on what appears to be a simultaneous basis. The speed at which the system components - both hardware and software - operate allows the system to switch from one active user to another, doing all or part of each job until all work is completed. The speed may be so great that each user believes that he or she is the only one using the system. The purpose for which one person uses the system may be totally unrelated to that of others. The system resources are shared by all.

There are three kinds of time-sharing systems in use today:

- General purpose systems. which support several programming languages and allow users to create and run their own programs;
- Systems in which a wide variety of programs are available for execution but cannot be modified by users;
- Systems in which all programs are related to one major application, and users merely provide input and request output.

In parctice a time-sharing system may be a combination of these, with the major application having first priority. The distinguishing characteristic among the three systems is the degree of user indepence provided.

From the point of view of Databases, Structured Query Language (SQL), allows real-time interactive users to access data by entering SQL statements; and programmers include SQL statements in application programs for the same purpose.

In a real-time environment, networks involve a communications or front- end processor which serves as a communication control unit in a system that must support many users and/or heavy traffic on the data-

communication network.

In a data-communication system, any I/O device at the end of a communication channel is called a terminal. More specifically, since the device is located at a point other than where the main computer is, it is called a **remote terminal.**

In real-time systems, speed of transmission is important.

6.8 PROBLEMS OF REAL - TIME SCHEDULING

When procedures are designed for the processing of real-time systems their operations are usually arranged so that all contigency handling has been preplanned and is therefore built in. In this way, any requirements for the treatment of exceptional cases or other error handling have been analysed in advance, and the corrective measures to be taken in the case of their occurrence have already been plannned and constructed. This style of operation may be termed 'off-line', for convenience; and this framework and approach may be considered as a contrast to those which employ adaptation strategies on the run - such procedures may similarly be styled as 'on-line'. The latter is self-evident in 'expert systems' and systems posssessing artificial intelligence.

'Off-line' maintenance of real-time systems is evident, for instance like in a local reservation system for railways where the system can be interrupted for some time and maintenance carried out. On the other hand, an expert system following a satellites journey in space may need 'on-line' maintenance.

Activity E

Mention three more examples each of 'off-line' and 'on-line' real-time systems with a brief description of its function.

.....

.....

.....

.....

6.9 SUMMARY

Effective application of computers is heavily dependent on software. Both system programs and application programs are required.

More sophisticated computers and operating systems that could handle job-to-job transitions were developed as time passed. Job-control statements were used to tell the computer what to do next in batch or stacked-job processing. On-line direct-access systems were developed to process input transactions in random order, as they were received, and to assist individual users in problem solving.

In a multitasking system, more than one program at a time resides in primary storage. The processor executes instructions from one program, then instructions from another program, and so on. Keeping the processor busy in this fashion helps to maximise the amount of useful work that can be accomplished.

In a multiprocessing system, instructions from the same program or from different programs are executed simultaneously on two or more processors. Today's personal computer systems as well as large system configurations may have multiprocessing capabilities. A real-time system provides output fast enough to satisfy any user requirements. It can be used to control real-life activity. It can be used to control real-life activity. Time - sharing allows several users to interact with a system on what appers to be a simultaneous basis. The system resources are shared amongst them.

In a virtual-machine environment, each user is able to use a total computer system -both hardware and softwae - that does not actually exist, but seems to. Many unrelated virtual machines running different jobs can be active on the system at a time.

6.10 SELF-ASSESSMENT EXERCISES

1) Explain in detail, with examples, what are the characteristics which go into the making of a high-quality system/program.

2) What are the various methods which can be selected for system implementation? Try to include in your answer the on-line system implementation suitability/unsuitability of each of the methods you describe.

3) What are the features of a language which should be considered for its selection in the design of a system? Explain your answer with special reference to on-line systems.

4) Explain what is system software and its significance in on-line system design.

5) Explain how problems in real-time scheduling are handled.

6) Explain and distinguish the following concepts with reference to their use in real-time systems.

   i) multiprocessing

   ii) time sharing

7) Explain the concepts of 'synchronous' and 'asynchronous' system software.

6.11 FURTHER READINGS

Szlanko, J. (Ed.)1986. *Real-time Programming: Proceedings of the 14th IFAC/IFIP Workshop,* Lake Balaton, Humgary, 26-28 May 1986; International Federation of Automatic Control of Pergammon Press.

Bohl, Marilyn, 1990. *Essentials of Information Processing*; Second Edition, Macmillan Publishing Company; New York and Collier Macmillan Publishers: London.

Core,Marvin R. and John W. Stubbe, 1987. *Computers and Information Systems;* Second Edition, McGraw-Hill International: New York.