UNIT 2 SYSTEM CONTROL AND QUALITY ASSURANCE

Structure

2.0 INTRODUCTION

The amount and complexity of software developed today stagger the imagination. Software development strategies have not come up with the standard and because of this software products do not meet the application objectives. Subsequently control must be developed to ensure a quality product. Basically quality assurance is the review of software products and its related documentation for completeness, correctness, reliability and maintainability, This unit deals with various issues involved in the quality assurance, testing and system control.

2.1 OBJECTIVES

The objectives of this unit are as follows:

- To highlight various issues involved in quality assurance, testing and system control.
- To point out the role of quality assurance in various stages of a Software Development Life Cycle.
- To illustrate the significance of various levels of tests and their different functions.
- To explain the role played by system control in a computer system.

2.2 QUALITY ASSURANCE IN SOFTWARE LIFE CYCLE

The software life cycle includes various stages of development and each stage has the goal of quality assurance. Steps taken in this regard are summarised below:

2.2.1 Quality Factors Specifications

The goal of this stage is to describe various factors mainly responsible for quality of the proposed system. They are as follows:

(i) Correctness: The extent to which a program meets system specifications and user objectives.

(ii) Reliability: The degree to which the system performs its intended functions overa time.

(iii) Efficiency: Computer resources required by a program to perform a particular function.

(iv) Usability: The efforts required to understand and operate a system.

(v) Maintainability: The ease with which the program errors are detected and removed.

(vi) Testability: The effort required to test a program to ensure its correct performance.

(vii) Portability: The ease of transporting a program from one hardware configuration to another.

(viii) Accuracy: The required precision in input, editing, computations, and output.

(ix) Error tolerance: Error detection and correction versus error avoidance.

(x) Expandability: Ease of expanding the existing database..

(xi) Access control and audit: Control of access to the system and the extent to which that access can be audited.

(xii) Communicativeness: Usefulness and effectiveness of the inputs and outputs of the system.

2.2.2 Software Requirements Specifications

The quality assurance goal of this stage is to generate the requirements document that helps in providing technical specifications lor developing the soflware.

2.2.3 Software Design Specifications

In this stage, the software design document defines the overall architecture of the software that provides the functions and features given in the software requirements document.

2.2.4 Software Testing and Implementation

The quality assurance goal of the testing phase is to'ensure that completeness and accuracy of the system and minimize the retesting process. In the implementation phase, the goal is to provide a logical order for the creation of the modules and, in turn, the creation of the system.

2.2.5 Maintenance and Support

This phase provides the necessary software development for the system to continue to comply with the original specifications. The quality assurance goal is to develop a procedure for correcting errors and enhancing software.

2.3 LEVELS OF QUALITY ASSURANCE

Analysts use three levels of quality assurance: testing, verification with validation and certification.

2.3.1 Testing

System testing is quite expensive and time consuming process. The common view of testing held by users is that it is performed to prove that program is error free. But this is quite difficult since the analyst cannot prove that softwarc is frce from all sort of errors.

Therefore the most useful and practical approach is with the understanding that testing is the processing of executing a program with the explicit intention of finding errors, that is, making the program fail. A successful test, then, is one that finds an error.

2.3.2 Verification with Validation

Like testing, verification is also intended to find errors. It is performed by executing a program in a simulated environment. Validation refers to the process of using software in a live environment to find errors.

When commercial systems are developed with the main aim of distributing them to dealers for sale purposes, they first go through verification, sometimes called alpha testing. The feedback from the validation phase generally brings some changes in the software to deal with errors and failures that are uncovered. Then a set of user sites is selected for putting the system into use on a live basis. These beta test sites use the system in day-to-day activities; they process live transactions and produce normal system output. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may bring more failures and the need for still more changes.

2.3.3 Certification

The last level of quality assurance is to certify that the software package developed conforms to standards. With a growing demand for purchasing ready-to-use software, importance of certification has increased. A package that is certified goes through a team of computer specialists who test, review and determine how well it meets the user's requirements and vendor's claims. Certification is issued only if the package is successful in all the tests. Certification, however, does not mean that it is the best package to adopt. I only attests that it will perform what the vendor claims.

2.4 DESIGN OBJECTIVES: RELIABILITY AND MAINTENANCE

The two operational design objectives continually sought by developers are systems reliability and maintainability. This section will discuss about the importance of these objectives and ways to achieve them.

2.4.1 Designing Reliable Systems

A system is said to be reliable if it does not produce dangerous or costly failures during its normal use. The definition recognises that systems may not always be used according to the designer's expectation. There are changes in the ways users use a system and also in business operations. However, there are steps analysts can follow to ensure that the system is reliable at the installation stage and its reliability will continue even after implementation.

There are two levels of reliability. The first level shows that the system is meeting the right requirements. This is possible only if a thorough and effective determination of systems requirements was performed by the analyst. A careful and thorough systems study is required for this aspect of reliability. The second level of systems reliability involves the actual working of the system delivered to the user. At this level, systems reliability is interwoven with software engineering and development.

Reliability has three approaches shown in the Table 1.

Table **1** : Approaches to Reliability

Sl. No.	*Approach*	*Description*	*Example*
1.	Error avoidance	Prevents errors from , occuring in the software	It is impossible in large systems.
2.	Error detection and correction	Recognises errors when they are encountered and corrects the error or the effect of the error so tat system does not fail.	Traps and modifies illegal arithmetic steps : Compensates for unexpected data values.
3.	Error Tolerance	Recognises errors when they occur, but enables the system to keep running through degraded performance.	Shuts down part of the system. Does not perform some processing but keeps the system operational.

2.4.2 Designing Maintainable Systems

When the systems are installed, they are normally used for a considerable period. The average life of a system is generally 4 to 6 years, with the oldest applications often in use for over 10 years. However, this period of constant use brings with it the used to contimually maintain the systcm. When system is fully implemented, analyst must take precautions to ensure that the need for mainlcnance is controlled through design and testing and the ability to perform it is provided through proper design practices.

Check Your Progress 1

1. List out various factors which arc responsible for the quality of a system.

....

....

....

....

....

2. Explain briefly the different levels of quality assurance.

....

....

....

....

....

3. Explain briefly the importance of system reliability.

....

....

....

....

....

....

2.3.1 Testing

System testing is quite expensive and time consuming process. The common view of testing held by users is that it is performed to prove that program is error free. But this is quite difficult since the analyst cannot prove that software is free from all sort of errors.

Therefore the most useful and practical approach is with the understanding that testing is the processing of executing a program with the explicit intention of finding errors, that is. making the program fail. A successful test, then, is one that finds an error.

2.3.2 Verification with Validation

Like testing, verification is also intended to find errors. It is performed by executing a program in a simulated environment. Validation refers to the process of using software in a live environment to find errors.

When commercial systems are developed with the main aim of distributing them to dealers for sale purposes, they first go through verification,sometimes called alpha testing. The feedback from the validation phase generally brings some changes in the software to deal with errors and failures that are uncovered. Then a set of user sites is selected for putting the system into use on a live basis. These beta test sites use the system in day-to-day activities; they process live transactions and produce normal system output. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may bring more failures and the need for still more changes.

2.3.3 Certification

The last level of quality assurance is to certify that the software package developed conforms to standards. With a growing demand for purchasing ready-to-use software, importance of certification has increased. A package that is certified goes through a team of computer specialists who test, review and determine how well it meets the user's requirements and vendor's claims. Certification is issued only if the package is successful in all the tests. Certification, however, does not mean that it is the best package to adopt. It only attests that it will perform what the vendor claims.

2.4 DESIGN OBJECTIVES: RELIABILITY AND MAINTENANCE

The two operational design objectives continually sought by developers are systems reliability and maintainability. This section will discuss about the importance of these objectives and ways to achieve them.

2.4.1 Designing Reliable Systems

A system is said to be reliable if it does not produce dangerous or costly failures during its normal use. The definition recognises that systems may not always be used according to the designer's expectation. There are changes in the ways users use a system and also in business operations. However, there are steps analysts can follow to ensure that the system is reliable at the installation stage and its reliability will continue even afler implementation.

There are two levels of reliability. The first level shows that the system is meeting the right requirements. This is possible only if a thorough and effective determination of systems requirements was performed by the analyst. A careful and thorough systems study is required for this aspect of reliability. The second level of systems reliability involves the actual working of the system delivered to the user. At this level, systems reliability is interwoven with software engineering and development.

Reliability has three approaches shown in the Table 1.

Table 1 : Approaches to Reliability

Sl. No.	*Approach*	*Description*	*Example*
1.	Error avoidance	Prevents errors from occuring in the software	It is impossible in large systems.
2.	Error detection and correction	Recognises errors when they are encountered and corrects the error or the effect of the error so tat system does not fail.	Traps and modifies illegal arithmetic steps : Compensates for unexpected data values.
3.	Error Tolerance	Recognises errors when they occur, but enables the system to keep running through degraded performance.	Shuts down part of the system. Does not perform some processing but keeps the system operational.

2.4.2 Designing Maintainable Systems

When the systems are installed, they are normally used for a considerable period. The average life of a system is generally 4 to 6 years, with the oldest applications often in use for over 10 years. However, this period of constant use brings with it the used to contimually maintain the system. When system is fully implemented, analyst must take precautions to ensure that the need for maintenance is controlled through design and testing and the ability to perform it is provided through proper design practices.

Check Your Progress 1

1. List out various factors which are responsible for the quality of a system.

....

....

....

....

....

2. . Explain briefly the different levels of quality assurance.

....

....

....

....

....

3. Explain briefly the importance of system reliability.

....

....

....

....

....

2.5 MAINTENANCE ISSUES

Many studies at the private, University and government level have been conducted to learn about maintenance requirements for information systems. These studies reveal the following facts:

(a) From 60 to 90 per cent of the overall cost of software during the life of a syslem is spent on maintenance.

(b) Often maintenance is not done very efficiently.

(c) Software demand is growing at a faster rate than supply. Many programmers are spending more time on systems maintenance than on new software development.

Several studies of maintenance have examined the type of tasks performed under maintenance (Lientz and Swanson, 1980). Table 2 summarizes the broad classes of maintenance found in information systems environments.

Table **2:** Types of System Maintenance

Category	Activity	Relative Frequency
Corrective	Emergency fixes, routine debugging.	20%
Adaptive	Accommodation of changes to data and files and to hardware and system software.	20%
Perfective	User enhancement, improved documentation, recording for computational efficiency.	60%

2.6 MAINTAINABLE DESIGNS

The keys to reduce the need for maintcnance, while making it possible to do essential tasks more efficiently, are as follows:

(a) More accurately defining the user's requirements during systems development,.

(b) Making better systems documentation.

(c) Using proper methods of designing processing logic and communicating it to project team members.

(d) Utilising the existing tools and techniques in an effective way.

(e) Managing the syslems engineering processin a better and effective way.

Now it is clear that design is both a process and a product. The design practices followed for software has a great effect on the maintainability of a system. Good design practices produce a product that can be maintained in a better way.

2.7 TESTING PRACTICE AND PLANS

It should be clear in mind that the philosophy behind testing is to find errors. Test cases are devised with this purpose in mind. A test case is **a** set of data that the system will process as normal input. However, the data are created with the express intent of determining whether the system will process them correctly, For example, test cases for inventory handling should include situations in which the quantities to be withdrawn from inventory exceed, equal and are less than the actual quantities on hand. Each test case is designed with the intent of finding errors in the way the **system** will process it. There are two general strategies for testing software: Code testing and Specification testing. In code testing, the analyst develops that cases to execute every instructions and path in a program. Under specification testing, the analyst examines the program specifications and then writes test data to determine how the program operates under specific conditions. Regardless of which strategy the analyst follows, there are preferred practices to ensure that the testing is useful. The levels of tests and types of test data, combined with testing libraries, are important aspects of the actual test process.

2.3 LEVELS OF TESTS

Systems are not designed as entire Systems nor are they tested as single systems. The analyst must perform both unit and system testing.

2.8.1 Unit Testing

In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing gives stress on the modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided. For example, a hotel information system consists of modules to handle rcscrvations; guest checkin and checkout; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing. For each, it provides the ability Io enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit testing should exercise each condition and option.

Unit testing can be performed from the bottom up, starting with smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perfonn the way it will when embedded within the larger system.

2.8.2 System Testing

The important and essential part of the system development phase, after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of errors in a newly designed system depend on some usual factors like communication between the user and the designer; the programmer's ability to generate a code that reflects exactly the systems specifications and the time frame for the design.

Theoretically, a newly designed system should have all the parts or sub-systems are in working order, but in reality, each sub-system works independently. This is the time Io gather all the subsystem into one pool and test the whole system to determine whether it meets the user requircmcnts. This is the last change to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits.

Testing is an important function to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

System testing consists of the following five steps:

- Program testing
- String testing
- System testing
- System documentation
- User acceptance testing

Program Testing

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly wilh olher programs. It is the responsibility of a programmer to have an error free program. At the time of testing the system, there exists two types of errors that should be checked. These errors are syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields out of range items, and invalid combinations. Since the logical errors are not detected by compiler, the programmer must examine the output carefully to detect them.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of the instructions, must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points.

String Testing

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each part of the system is tested against the entire module with bolh test and live data before the whole system is ready to be tested.

System Testing

System testing is designed to uncover weaknesses that were not found in earlier tests. This includes forced system failure and validation of total system as it will be implemented by its user in the operational environment. Under this testing, generally we take low volumes of transactions based on live data. This volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. When we see that the proposed system is successful in the test, the old system is discontinued.

System Documentation

All design and test documentation should be well prepared and kept in the library for future reference. The library is the central location for maintenance of the new system.

User Acceptance Testing

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the syslem. After that a comprehensive test report is prepared. This report shows the system's tolerance, performance range, error rate and accuracy.

2.9 SPECIAL SYSTEMS TESTS

There are other six tests which fall under special category. They are described below :

(i) Peak **Load** Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.

(ii) **Storage** Testing: It determines the capacity of the system to store transaction data on a disk or in other files. For example, verify documentation statements that the **system** will store 10,000 records of 400 bytes length on a single flexible disk.

(iii) Performance Time Testing: It determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.

(iv) Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.

(v) **Procedure Testing: It determines the clarity of documentation on operation and use of** system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer,

(vi) Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

Check Your Progress 2

1. What are the different types of system maintenance? Explain them briefly.

....

....

....

....

....

2. Describe briefly about 'Unit Testing'.

....

....

....

....

....

3. What do you know about various special systems tests? Explain briefly.

....

....

....

....

....

2.10 DESIGNING TEST DATA

The proper designing of test data is as important as the test itself. E lest dam as input are not valid or representation of the data to bc provided by the user, then the reliability of the output is doubtful. Test data may be live or artificial. The live data is hat which is actually extracted from the users' files. After a system is partially constructed, the programmers or analysts ask the users to key in a set of data from their normal activities. It is difficult to obtain live data in sufficient amount to conduct extensive testing.

The artificial test dam is created solely for test purposes. Properly created artificial data should provide all combinations of values and formats and make it possible to test all logic and control paths through the program. Unlike live data, which are biased toward typical values, artificial data provide extreme values for testing the limits of the proposed system.

2.11 SYSTEM CONTROL

In a computer system, controls essentially mean the extent to which the system is secure against human errors, machine malfunction or deliberate mischief. The amount of control to be applied can vary depending upon the importance, critically and volume of the output. Depending on the nature of the system or the amounts of money at stake, the designer will need to build in different type of controls within system procedures, programs and operations.

2.11.1 Objective of System Control

Control mechanisms are designed to achieve the following objectives:

(a) Accuracy: The system should provide information and reports which are accurate in all respect.

(b) Reliability: The system should continue to function as it is designed to function. It should not break down due to malfunction of equipments.

(c) Security: Files and programs in the system must be secured against accidental damage or loss. Procedures must be established to restrict access to data by authorised user only.

(d) Efficiency: Efficiency of the system will essentially mean performing the tasks in the best manner and producing output of highest quality with least efforts.

(e) Audit: System controls should cover the aspect of auditing procedures. Facilities should be designed so that a transaction can be traced from its creation to its final use. This aspect of control assumes added significant in case of on-line systems where written documents may not exist for some transactions.

(f) Adherence to organisation policies: System controls should not conflict with organisation policies and must promote such policies. For example, by ensuring that payroll is produced accurately and by a specified time, the organisation objective of paying all employees in time will be promoted.

It may not always be possible to provide adequate controls in the system to meet the above objectives fully. Sometimes, it may happen that too many controls may adversely affect the performance of the system. On the other hand, lack of adequate control may create chaos and dissatisfaction. It is, therefore, a question of trade-off between the control and their objectives and as stated earlier, the decisions will depend on the nature of the system and criticality of its functions.

2.11.2 Types of Control

Two types of controls which affect the operation of a system are:

- External control
- Internal control

External controls to a system, as the name implies are laws, regulations, procedures and policies outside the scope of the system which affect the operations of the system.

Internal controls are basically plans, procedures, guidelines, rules and checks under which the system must function. Much of these internal controls will be specified as a part of system design, but some of these may be internal controls of the organisation - with or without computcrisation.

2.12 AUDIT TRAIL

In on-line systems, unlike batch environments, there may not be copies of input source documents to fall back on if the system fails during processing. It is also possible for on-line users to sign on to a system, make changes in data already stored in files and sign off again without leaving a visible clue as to what happened. Unless the systems analyst develops an audit trail, no such protection exists in on-line systems.

Audit trail is the path which a transaction traces through a data processing system from source documents to summary reports. In other words, it refers to the facilities or procedures which allow a transaction to be traced through all stages of data processing beginning with its appearance on a source document and ending with its transformation into information on a final output document. The audit uail contains complete derails such as reference numbers, dates, names which are recorded in files, ledgers and journals so that trailing of these records to source documents becomes easier. It is generally available in manual accounting system. But in computerised system, it is generally not available unless the system is specially designed to do so.

Audit trails are not primarily for the use of auditors. Rather they are quite important tools that are designed to help the management. The auditors use these tools which management has found necessary for internal purposes.

Check Your Progress 3

1. List out the objectives of system control.

....

....

....

....

....

2. What is the difference between internal and external control?

....

....

....

....

....

3. Explain briefly about 'Audit Trail'.

....

....

....

....

....

2.13 SUMMARY

It is thus seen that quality assurance, testing and system control play a vital role and each of them contribute to the devefopment of an efficient software. Quality assurance has to be implemented at every stage of the software development life cycle. Implementation at each lcvcl thus paves way for a better step in the next stage. Both the system testing and the unit testing should be done so as to avoid the occurrence of error during implementation stage. Using Audit trail all the transactions occurring during a period of time can be tracked.

2.14 MODEL ANSWERS

Check Your Progress 1

1. Various factors are as follows:

(i) Correctness	(ii) Reliability	(iii) Efficiency	(iv) Usability
(v) Maintainability	(vi) Testability	(vii) Portability	(viii) Accuracy
(ix) Error tolerane	(x) Expandability	(xi) Access control and audit	
(xii) Communicativeness			

2. Different levels of quality assurance are:

   (i) Testing

   (ii) Verification with validation

   (iii) Certification

3. A system is said to be reliable if it does not produce dangerous or costly failures during its normal use. There are two levels of reliability. The first level shows that system is meeting the right requirements. This is possible only if a thorough and effective determination of system requirement, was properly studied by the analyst. The second level of system reliability involves the actual working of the system delivered to the user. At this level, system's reliability is interwoven with software engineering and development.

Check Your Progress 2

1. Different types of system maintenance are:

   (a) Corrective

   (b) Adaptive

   (c) Perfective

2. In unit testing, the analyst tests the programs making up a system. Unit testing gives stress on the modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided.

3. Various special system tests are:

(i) Peak load test	(ii) Storage testing	(iii) Performance time testing
(iv) Recovery testing	(v) Procedure testing	(vi) Human factors testing

Check **Your** Progress **3**

1. Objective of system control arc as under:

(i) Accuracy	(ii) Reliability	(iii) Security	(iv) Efficiency
(v) Audit	(vi) Adherence to organisation policies		

2. Internal controls: They are basically plans, procedures, guidelines, rules and checks under which the system must function.

   External controls: They me mainly laws, regulations, procedures and policies outside the scope of the system which affect the operations of the system.

3. Audit trail is the name given to the facility to trace individual transactions through a system from source to completion.