SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT-I- SOFTWARE TESTING – SBS1604

COURSE OBJECTIVES:

- To study fundamental concepts in software testing.
- To discuss various software testing issues and solutions in software unit test, integration and system testing.
- To expose the advanced software testing topics, such as object-oriented software testing methods.
- Finding defects which may get created by the programmer while developing the software.

COURSE OUTCOMES:

At the end of this course student will:

- List a range of different software testing techniques and strategies and be able to apply specific(automated) unit testing method to the projects.
- Distinguish characteristics of structural testing methods.
- Demonstrate the integration testing which aims to uncover interaction and compatibility problems as early as possible.
- Discuss about the functional and system testing methods.
- Demonstrate various issues for object-oriented testing.

UNIT 1

Introduction: Software testing – Role of software testing – Three step process to becoming a world class testing organization - A structural approach to testing – Test strategy – methods for developing test strategy

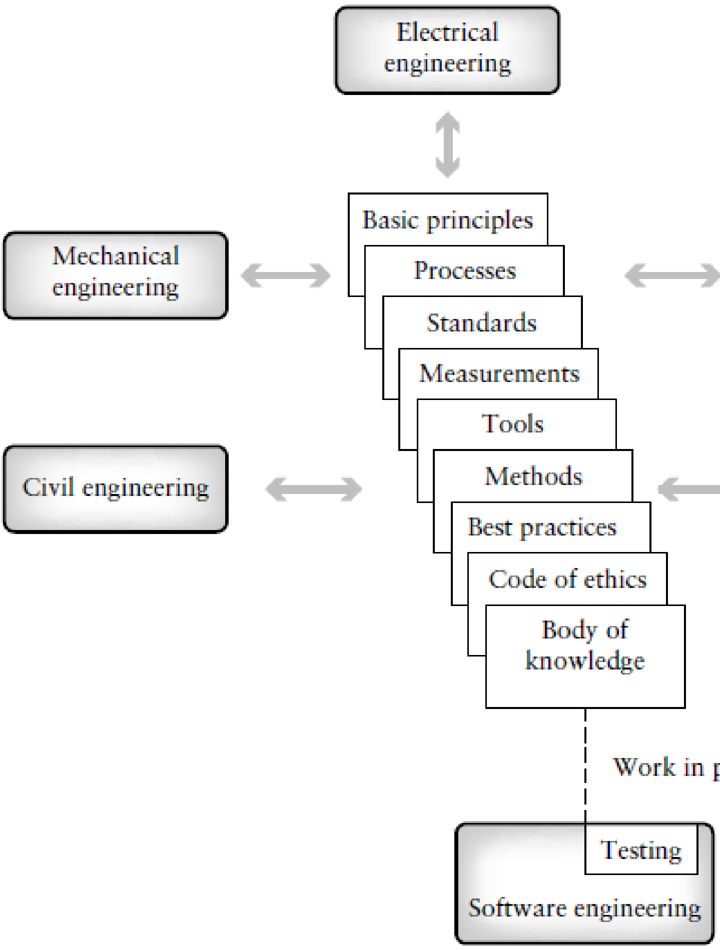
Introduction: Software testing

The Evolving Profession of Software Engineering

- This is an exciting time to be a software developer.
- Software systems are becoming more challenging to build.
- They are playing an increasingly important role in society.
- People with software development skills are in demand.
- New methods, techniques, and tools are becoming available to support development and maintenance tasks.
- Because software now has such an important role in our lives both economically and socially, there is pressure for software professionals to focus on quality issues.
- Poor quality software that can cause loss of life or property is no longer acceptable to society.
- Failures can result in catastrophic losses.
- Conditions demand software development staff with interest and training in the areas of software product and process quality.
- Highly qualified staff ensure that software products are built on time, within budget, and are of the highest quality concerning attributes such as reliability, correctness, usability, and the ability to meet all user requirements.
- In response to the demand for high-quality software, and the need for well-educated software professionals, there is a movement to change the way software is developed and maintained, and the way developers and maintainers are educated.
- The profession of software engineering is slowly emerging as a formal engineering discipline.
- As a new discipline, it will be related to other engineering disciplines, and have associated with it a defined body of knowledge, a code of ethics, and a certification process.
- The education and training of engineers in each engineering discipline are based on the teaching of related scientific principles, engineering processes, standards, methods, tools, measurement, and best practices as shown in the Figure below.
- As a reflection of the movement toward a software engineering profession, and these educational needs, the IEEE Computer Society and the Association of Computing Machinery (ACM), the two principal societies for software professionals, have appointed joint task forces.
- The goals of the task force teams are to define a body of knowledge that covers the software engineering discipline, to discuss the nature of education for this new profession, and to define a code of ethics for the software engineer.
- Foreseeing the emergence of this new engineering discipline, some states are already preparing licensing examinations for software engineers.
- This text is based on the philosophy that software development should be viewed and taught as an engineering discipline and that quality in both the process and the product are of prime importance to professionals in this field.

Using an engineering approach to software development implies that:

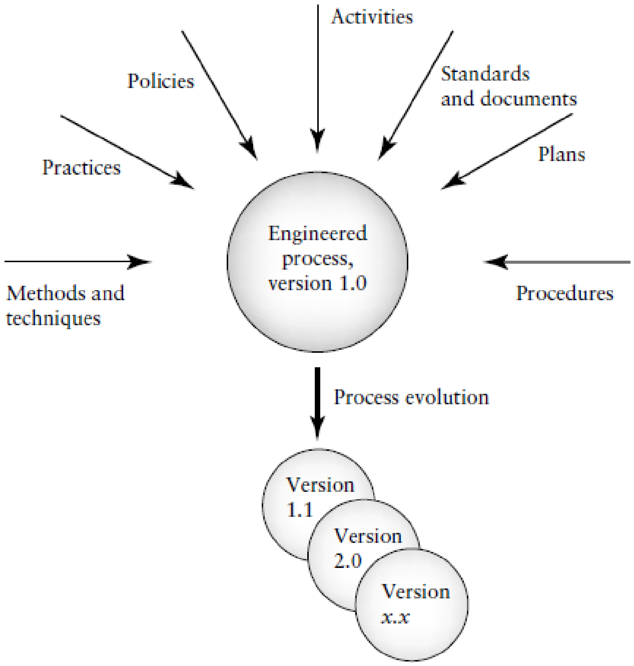
- The development process is well understood
- Projects are planned
- Life cycle models are defined and adhered to
- Standards are in place for product and process
- Measurements are employed to evaluate product and process quality
- Components are reused
- Validation and verification processes play a key role in quality determination
- Engineers have proper education, training, and certification



- This text aims to support the education of a software professional called a test specialist.
- A test specialist is one whose education is based on the principles, practices, and processes that constitute the software engineering discipline, and whose specific focus is on one area of that discipline—software testing.
- A test specialist who is trained as an engineer should know test-related principles, processes, measurements, standards, plans, tools, and methods, and should learn how to apply them to the testing tasks to be performed.
- This text aims to educate the reader in the testing discipline.
- Testing concepts, instead of being presented as an isolated collection of technical and managerial activities will instead be integrated within the context of a quality testing process that grows incompetency and uses engineering principles to guide improvement growth.
- In this way, all of the elements of the testing discipline emerge incrementally and allow the tester to add knowledge and skills that follow a natural evolutionary pattern.
- The integrating framework for presenting testing concepts is the Testing Maturity Model.

The Role of Process in Software Quality

- The need for software products of high quality has pressured those in the profession to identify and quantify quality factors such as usability, testability, maintainability, and reliability, and to identify engineering practices that support the production of quality products having these favourable attributes.
- Among the practices identified that contribute to the development of high-quality software are project planning, requirements management, development of formal specifications, structured design with use of information hiding and encapsulation, design and code reuse, inspections and reviews, product and process measures, education and training of software professionals, development and application of CASE tools, use of effective testing techniques, and integration of testing activities into the entire life cycle.
- In addition to identifying these individual best technical and managerial practices, software researchers realized that it was important to integrate them within the context of a high-quality software development process.
- Process, in the software engineering domain, is the set of methods, practices, standards, documents, activities, policies, and procedures that software engineers use to develop and maintain a software system and its associated artifacts, such as project and test plans, design documents, code, and manuals.



- It also was clear that adding individual practices to an existing software development process in an ad hoc way was not satisfactory.
- The software development process, like most engineering artifacts, must be engineered.
- That is, it must be designed, implemented, evaluated, and maintained.
- As in other engineering disciplines, a software development process must evolve consistently and predictably, and the best technical and managerial practices must be integrated systematically.
- Models such as the Capability Maturity Model (CMM)* and SPICE were developed to address process issues.
- These models allow an organization to evaluate its current software process and to capture an understanding of its state.
- Strong support for incremental process improvement is provided by the models, consistent with historical process evolution and the application of quality principles.

- The models have received much attention from the industry, and resources have been invested in process improvement efforts with many successes recorded.
- All the software process improvement models that have had wide acceptance in the industry are high-level models, in the sense that they focus on the software process as a whole and do not offer adequate support to evaluate and improve specific software development sub-processes such as design and testing.
- Most software engineers would agree that testing is a vital component of a quality software process, and is one of the most challenging and costly activities carried out during software development and maintenance.
- Despite its vital role in the production of quality software, existing process evaluation and improvement models such as the CMM, Bootstrap, and ISO-9000 have not adequately addressed testing process issues.
- The Testing Maturity Model (TMM), as described throughout this text, has been developed at the Illinois Institute of Technology by a research group headed by the author, to address deficiencies in these areas.

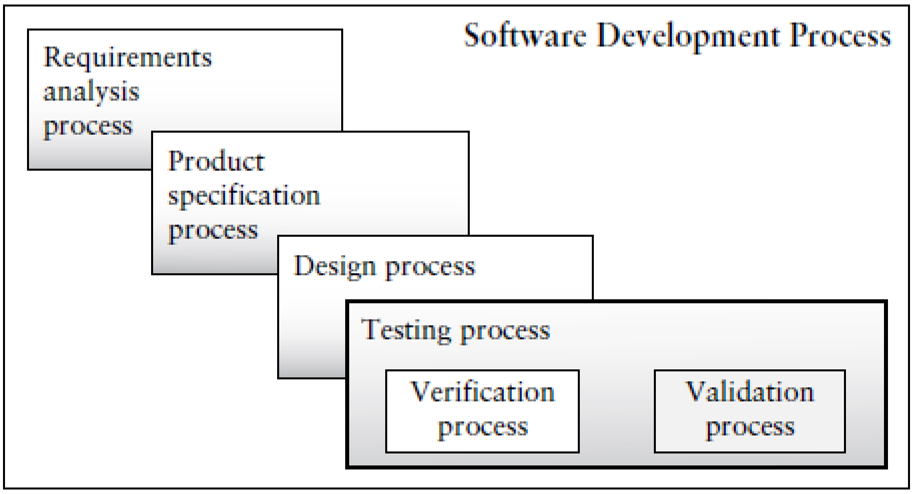
Testing as a Process

- The software development process has been described as a series of phases, procedures, and steps that result in the production of a software product.
- Embedded within the software development process are several other processes including testing.
- Some of these are shown in Figure
- Testing itself is related to two other processes called verification and validation as shown in Figure

Validation is the process of evaluating a software system or component during, or at the end of, the development cycle to determine whether it satisfies specified requirements.

Validation is usually associated with traditional execution-based testing, that is, exercising the code with test cases.

Verification is the process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.


Software Development Process
Requirements analysis process
Product specification process
Design process
Testing process
Verification process
Validation process

Verification is usually associated with activities such as inspections and reviews of software deliverables.

Testing itself has been defined in several ways. Two definitions are shown below.

Testing is generally described as a group of procedures carried out to evaluate some aspect of a piece of software.

Testing can be described as a process used for revealing defects in software, and for establishing that the software has attained a specified degree of quality concerning selected attributes.

Note that these definitions of testing are general. They cover both validation and verification activities and include in the testing domain all of the following: technical reviews, test planning, test tracking, test case design, unit test, integration test, system test, acceptance test, and usability test. The definitions also describe testing as a dual-purpose process—one that reveals defects, as well as one that is used to evaluate quality attributes of the software such as reliability, security, usability,
and correctness. Also note that testing and debugging, or fault localization, are two very different activities. The debugging process begins after testing has been carried out and the tester has noted that the software is not behaving as specified.

Debugging, or fault localization is the process of (1) locating the fault or defect, (2) repairing the code, and (3) retesting the code.

Testing as a process has economic, technical, and managerial aspects. Economic aspects are related to the reality that resources and time are available to the testing group on a limited basis. In fact, complete testing is in many cases not practical because of these economic constraints. An organization must structure its testing process so that it can deliver software on time and within budget, and also satisfy the client's requirements. The technical aspects of testing related to the techniques, methods, measurements, and tools used to insure that the software under test is as
defect-free and reliable as possible for the conditions and constraints under which it must operate. Testing is a process, and as a process, it must be managed. Minimally that means that an organizational policy for testing must be defined and documented. Testing procedures and steps must be defined and documented. Testing must be planned, testers should be trained, the process should have associated quantifiable goals that can be measured and monitored. Testing as a process should be able to evolve to a level where there are mechanisms in place for making continuous improvements.

An Overview of the Testing Maturity Model

Several important test-related issues have emerged from the previous discussion.

We have learned that

1. there is a demand for software of high quality with low defects;
2. process is important in the software engineering discipline;
3. software testing is an important software development sub-process;
4. existing software evaluation and improvement models have not adequately addressed testing issues.

An introduction to the Testing Maturity Model is now presented to the reader as a framework for discussion of these issues, and as a means for addressing them. The model is discussed in more detail in later chapters of this text. The focus of the TMM is on testing as a process in itself that can be evaluated and improved.

In the testing domain possible benefits of test process improvement are the following:

- Smarter testers
- Higher quality software
- Ability to meet budget and schedule goals
- Improved planning
- Ability to meet quantifiable testing goals

Test process improvement is supported by the set of levels and maturity goals in the TMM. Achievement of the maturity goals results in incremental improvement of an organization’s testing process. The TMM Assessment Model supports test process evaluation. Section 1.3 gives the reader an overview of the set of levels and maturity goals. The levels and goals serve as guidelines for the organization of this text and define the sequence for the introduction of testing concepts. The development of version 1.0 of the TMM was guided by the work done on the Capability Maturity Model for software (CMM), a process improvement model that has received widespread support from the software industry in the United States. The CMM is classified architecturally as a staged process improvement model. This type of process improvement model architecture prescribes the stages that an organization must proceed through in an orderly fashion to improve its software development process. Other process improvement models can be described as having a continuous type of architecture, for example, the SPICE model. In this type of architecture, there is no fixed set of levels or stages to proceed through. An organization applying a continuous model can select areas for improvement from many different categories. The CMM has five levels or stages that describe an evolutionary pattern of software process maturity and serve as a guide for improvement. Each level has a set of Key Process Areas (KPA) that an organization needs to focus on to achieve maturity at that level. There are also key practices associated with each level that provide support for implementing improvements at that level. The CMM also has an assessment procedure that allows an organization to evaluate the current state of its software process and identify process strengths and weaknesses.

Other input sources to TMM development include Gelperin and Hetzel’s Evolution of Testing Model, which describes the evolution of the testing process in the industry over 40 years; Beizer’s testing model, which describes the evolution of the individual tester’s thinking; and the Software Testing Practices Survey Report, which identifies best test practices in industry as of 1993. More details relating to these items as well as the TMM maturity goals and the TMM Assessment Model are found in later chapters of this text.

TMM Levels

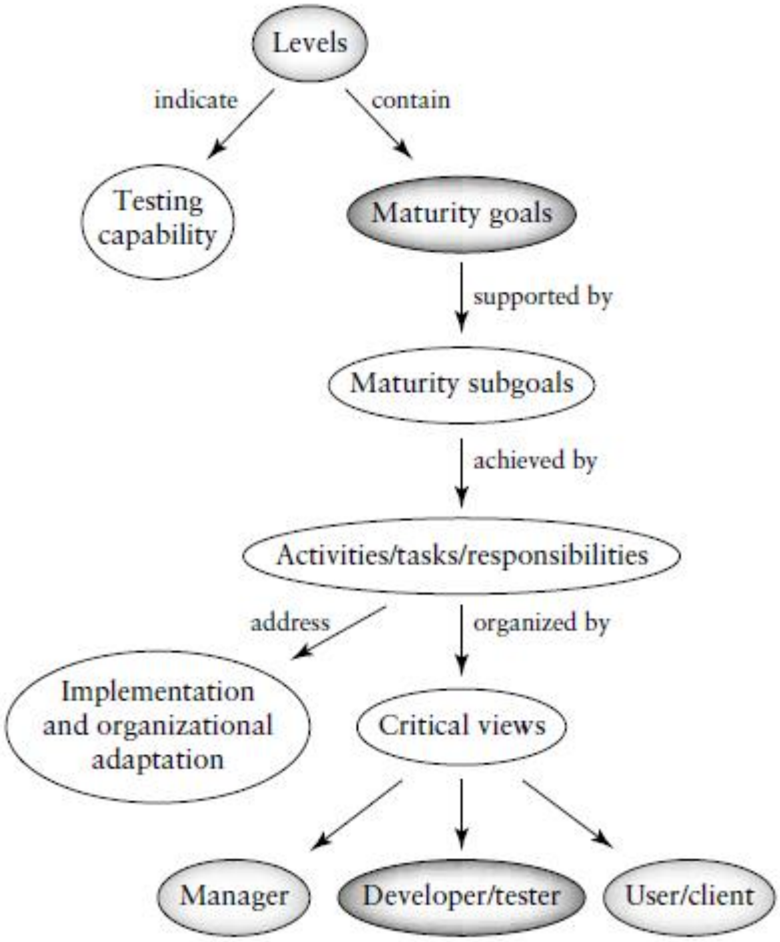
As in the case of the CMM, the TMM also follows what is called a staged architecture for process improvement models. It contains stages or levels through which an organization passes as its testing process evolves from one that is ad hoc and unmanaged to one that is managed, defined, measured, and optimizable. The internal structure of the TMM is rich in testing practices that can be learned and applied in a systematic way to support a quality testing process that improves in incremental steps. There are five levels in the TMM that prescribe a maturity hierarchy and an evolutionary path to test process improvement. The characteristics of each level are described in terms of testing capability organizational goals, and roles/responsibilities for the key players in the testing process, the managers, developers/testers, and users/clients.

Each level except for level 1 has a structure that consists of the following:

• A set of maturity goals. The maturity goals identify testing improvement goals that must be addressed to achieve maturity at that level. To be placed at a level, an organization must satisfy the maturity goals at that level. The TMM levels and associated maturity goals are shown in Figure

• **Supporting maturity subgoals.** They define the scope, boundaries, and needed accomplishments for a particular level.

• **Activities, tasks, and responsibilities (ATR).** The ATRs address implementation and organizational adaptation issues at each TMM level.



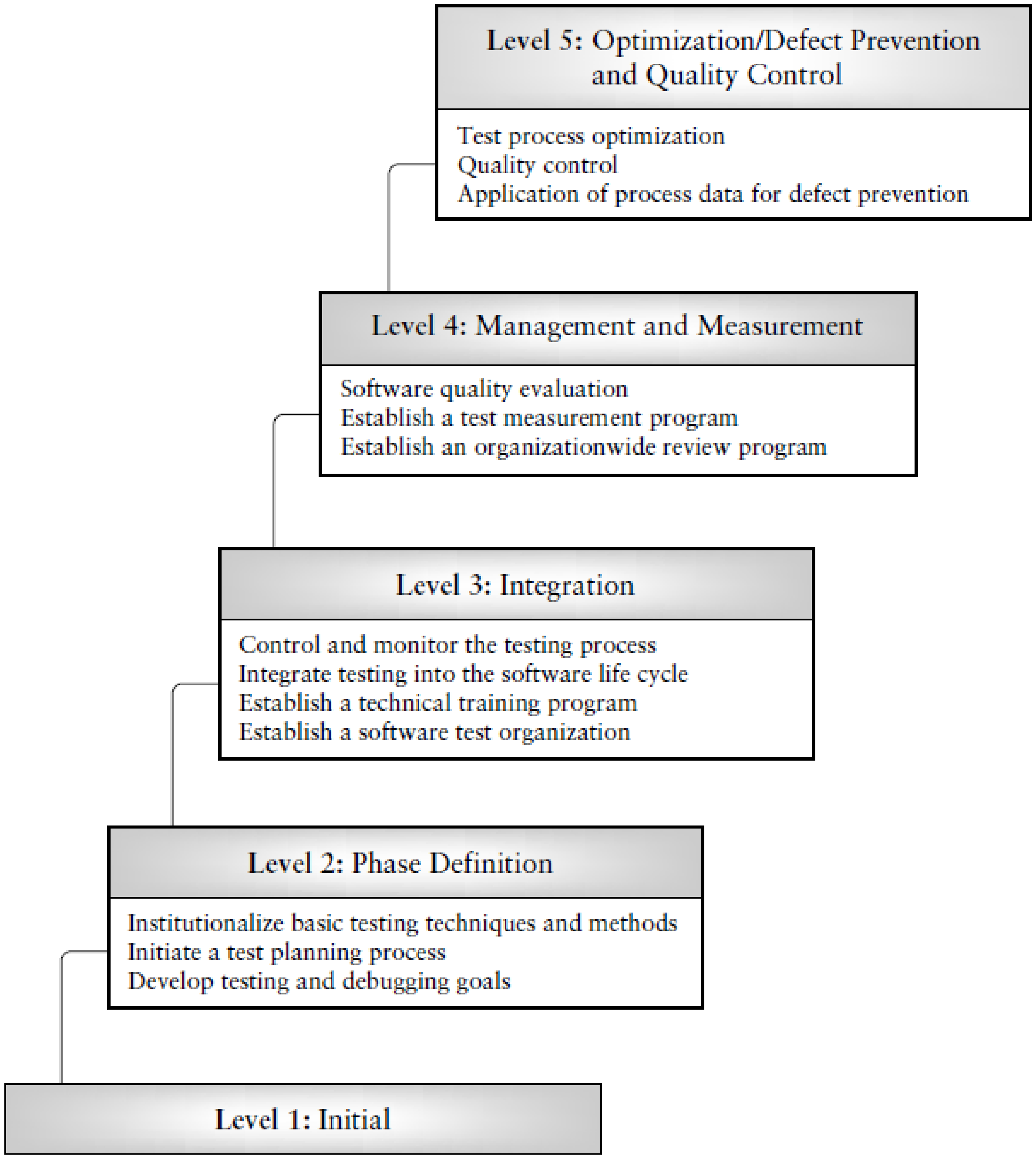
Supporting activities and tasks are identified, and responsibilities are assigned to appropriate groups. Figure 1.4 illustrates the TMM level structure. Each maturity goal at each TMM level is supported by a set of maturity subgoals. The maturity subgoals are achieved through a group of activities and tasks with responsibilities (ATR). Activities and tasks are defined in terms of actions that must be performed at a given level to improve testing capability; they are linked to organizational commitments. Responsibilities are assigned for these activities and tasks to three groups that TMM developers believe represent the key participants in the testing process: managers, developers/ testers, and users/clients. In the model, they are referred to as “the three critical views (CV).” The definition of their roles is essential in developing a maturity framework. The manager’s view involves commitment and the ability to perform activities and tasks related to improving testing capability. The developer/tester’s view encompasses the technical activities and tasks that, when applied, constitute quality testing practices. The user’s or client’s view is defined as a cooperating or supporting, view. The developers/ testers work with client/user groups on quality-related activities and tasks that concern user-oriented needs. The focus is on soliciting client/ user support, consensus, and participation in activities such as requirements analysis, usability testing, and acceptance test planning.

The maturity goals at each level of the TMM are shown in Figure 1.5. They are fully described in published papers and are also listed below along with a brief description of the characteristics of an organization at each TMM level [2–6]. The description will introduce the reader to the evolutionary path prescribed in the TMM for test process improvement.
Additional details are provided in subsequent text chapters.

Level 1—Initial: (No maturity goals)
At TMM level 1, testing is a chaotic process; it is ill-defined, and not distinguished from debugging. A documented set of specifications for software behavior often does not exist. Tests are developed in an ad hoc way after coding is completed. Testing and debugging are interleaved to get the bugs out of the software. The objective of testing is to show the software works (it is minimally functional) [1,5]. Software products are
often released without quality assurance. There is a lack of resources, tools, and properly trained staff. This type of organization would be at level 1 of the CMM.

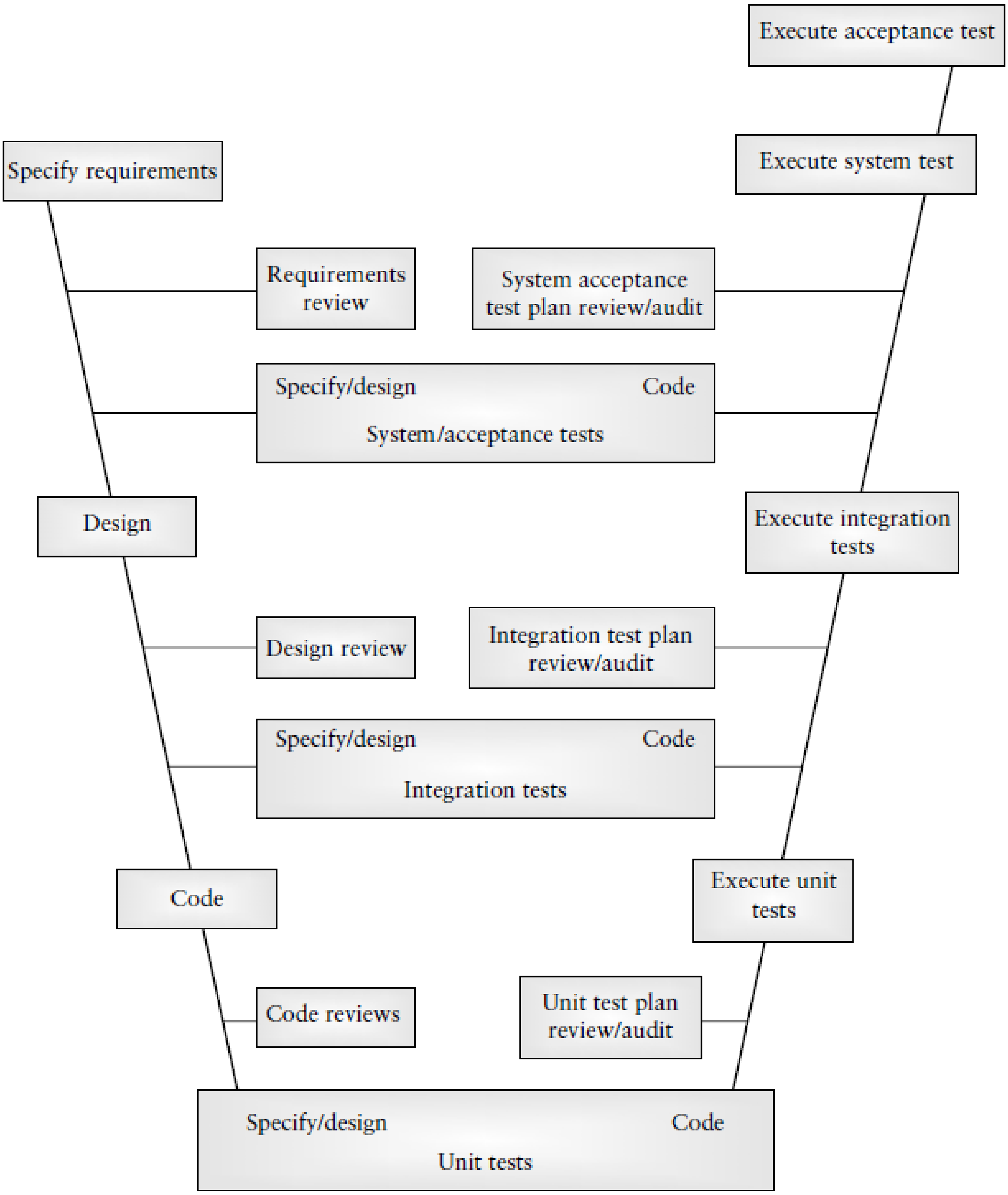
Level 2—Phase Definition: (Goal 1: Develop testing and debugging goals; Goal 2: Initiate a testing planning process; Goal 3: Institutionalize basic testing techniques and methods)
At level 2 of the TMM testing is separated from debugging and is defined as a phase that follows coding. It is a planned activity; however, test planning at level 2 may occur after coding for reasons related to the immaturity of the testing process. For example, there may be the perception at level 2, that all testing is execution based and dependent on the code; therefore, it should be planned only when the code is complete. The primary goal of testing at this level of maturity is to show that the software meets its stated specifications [2,5]. Basic testing techniques and methods are in place; for example, the use of black box and white box testing strategies, and a validation cross-reference matrix. Testing is multileveled: there are unit, integration, system, and acceptance levels. Many quality problems at this TMM level occur because test planning occurs late in the software life cycle. Also, defects are propagated from the requirements and design phases into the code. There are no review programs as yet to address this important issue. Postcode, execution-based testing is still considered the primary testing activity.

Level 3—Integration: (Goal 1: Establish a software test organization; Goal 2: Establish a technical training program; Goal 3: Integrate testing into the software life cycle; Goal 4: Control and monitor testing)
At TMM level 3, testing is no longer a phrase that follows coding but is integrated into the entire software life cycle. Organizations can build on the test planning skills they have acquired at level 2. Unlike level 2, planning for testing at TMM level 3 begins at the requirements phase and continues throughout the life cycle supported by a version of the V-model (see Section 8.7) [2]. Test objectives are established concerning the requirements based on user/client needs and are used for test case design. There is a test organization, and testing is recognized as a professional activity. There is a technical training organization with a testing focus. Testing is monitored to ensure it is going according to plan and actions can be taken if deviations occur. Basic tools support key testing activities, and the testing process is visible in the organization. Although organizations at this level begin to realize the important role of reviews in quality control, there is no formal review program and reviews do not as yet take place across the life cycle. A formal test measurement program has not yet been established to quantify a significant number of process and product attributes.


Level 5: Optimization/Defect Prevention and Quality Control
Test process optimization
Quality control
Application of process data for defect prevention
Level 4: Management and Measurement
Software quality evaluation
Establish a test measurement program
Establish an organizationwide review program
Level 3: Integration
Control and monitor the testing process
Integrate testing into the software life cycle
Establish a technical training program
Establish a software test organization
Level 2: Phase Definition
Institutionalize basic testing techniques and methods
Initiate a test planning process
Develop testing and debugging goals
Level 1: Initial

Level 4—Management and Measurement: (Goal 1: Establish an organization-wide review program; Goal 2: Establish a test measurement program; Goal 3: Software quality evaluation)

Testing at level 4 becomes a process that is measured and quantified. Reviews at all phases of the development process are now recognized as testing/quality control activities. They are a complement to execution-based tests to detect defects and to evaluate and improve software quality. An extension of the V-model as shown in Figure 1.6 can be used to support the implementation of this goal [6,7]. Software products are tested for quality attributes such as reliability, usability, and maintainability. Test cases from all projects are collected and recorded in a test case database for test case reuse and regression testing. Defects are logged and given a severity level. Some of the deficiencies occurring in the test process are due to the lack of a defect prevention philosophy, and the porosity of automated support for the collection, analysis, and dissemination of test-related metrics.



Level 5—Optimization/Defect Prevention/Quality Control: (Goal 1: Defect prevention; Goal 2: Quality control; Goal 3: Test process optimization)
Because of the infrastructure that is in place through achievement of the maturity goals at levels 1–4 of the TMM, the testing process is now said to be defined and managed; its cost and effectiveness can be monitored. At level 5, mechanisms are in place so that testing can be fine-tuned and continuously improved. Defect prevention and quality control are practiced. Statistical sampling, measurements of confidence levels, trustworthiness, and reliability drive the testing process. Automated tools support the running and rerunning of test cases. Tools also provide support for test case design, maintenance of test-related items, and defect collection and analysis. The collection and analysis of test-related metrics also have tool support. Process reuse is also a practice at TMM level 5 supported by a Process Asset Library (PAL).

TESTING FUNDAMENTALS

Initiating a Study of Testing

The study of software testing in this text begins with a description of essential test-related vocabulary items. Knowledge of these basic terms is essential to ensure that the discussions of testing concepts that follow are based on a common vocabulary that is widely accepted in academia and industry. A set of execution-based testing principles is also presented here to support test specialists. They provide a foundation for developing testing knowledge, acquiring testing skills, and developing an essential group of best practices. This introduction to the field of software testing concludes with a description of the role of the test specialist in a software development organization.

Basic Definitions

Below is a set of basic definitions for terms that will be used in this text. Additional definitions appear in subsequent chapters to aid in concept understanding. Many of the definitions used in this text are based on the terms described in the IEEE Standards Collection for Software Engineering [1]. The standards collection includes the IEEE Standard Glossary of Software Engineering Terminology, which is a dictionary devoted to describing software engineering vocabulary [2]. It contains working definitions of terms that are in use in both the academic and industrial worlds.

Where a definition has been directly adapted from an IEEE standards document a specific reference is given.

Errors

An error is a mistake, misconception, or misunderstanding on the part of a software developer.

In the category of developers we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a design notation, or a programmer might type a variable name incorrectly.

Faults (Defects)

A fault (defect) is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification.

Faults or defects are sometimes called “bugs.” Use of the latter term trivializes the impact faults have on software quality. Use of the term “defect” is also associated with software artifacts such as requirements and design documents. Defects occurring in these artifacts are also caused by errors and are usually detected in the review process.

Failures

A failure is the inability of a software system or component to perform its required functions within specified performance requirements [2].

During execution of a software component or system, a tester, developer, or user observes that it does not produce the expected results. In some cases a particular type of misbehaviour indicates a certain type of fault is present. We can say that the type of misbehaviour is a symptom of the fault. An experienced developer/tester will have a knowledge base of fault/symptoms/failure cases (fault models as described in Chapter 3) stored in memory. Incorrect behaviour can include producing incorrect values for output variables, an incorrect response on the part of a device, or an incorrect image on a screen. During development failures are usually observed by testers, and faults are located and repaired by developers. When the software is in operation, users may observe failures which are reported back to the development organization so repairs can be made. A fault in the code does not always produce a failure. In fact, faulty software may operate over a long period of time without exhibiting any incorrect behaviour. However, when the proper conditions occur the fault will manifest itself as a failure. Voas is among the researchers who discuss these conditions, which are as follows:

1. The input to the software must cause the faulty statement to be executed.
2. The faulty statement must produce a different result than the correct statement.
3. This event produces an incorrect internal state for the software.
4. The incorrect internal state must propagate to the output, so that the result of the fault is observable.

Software that easily reveals its' faults as failures is said to be more testable. From the testers point-of-view this is a desirable software attribute. Testers need to work with designers to insure that software is testable. There are other meanings assigned to the terms "testable" and "testability" that will be described later on in this chapter.

Test Cases

The usual approach to detecting defects in a piece of software is for the tester to select a set of input data and then execute the software with the input data under a particular set of conditions. In order to decide whether the software has passed or failed the test, the tester also needs to know what are the proper outputs for the software, given the set of inputs and execution conditions. The tester bundles this information into an item
called a test case.

A **test case** in a practical sense is a test-related item which contains the following information:

1. A set of test inputs. These are data items received from an external source by the code under test. The external source can be hardware, software, or human.
2. Execution conditions. These are conditions required for running the test, for example, a certain state of a database, or a configuration of a hardware device.
3. Expected outputs. These are the specified results to be produced by the code under test.

The above description specifies the minimum information that should be found in a test case and is based on the IEEE description for this item [2]. An organization may decide that additional information should be included in a test case to increase its value as a reusable object, or to provide more detailed information to testers and developers. As an example, a test objective component could be included to express test goals such as to execute a particular group of code statements or check that a given requirement has been satisfied. Developers, testers, and/or software quality assurance staff should be involved in designing a test case specification that precisely describes the contents of each test case. The content and its format should appear in test documentation standards for the organization.

Test

A test is a group of related test cases, or a group of related test cases and test procedures

A group of related tests is sometimes referred to as a test set. A group of related tests that are associated with a database, and are usually run together, is sometimes referred to as a test suite

Test Oracle

A test oracle is a document, or piece of software that allows testers to determine whether a test has been passed or failed.

A program, or a document that produces or specifies the expected outcome of a test, can serve as an oracle. Examples include a specification (especially one that contains pre- and postconditions), a design document, and a set of requirements. Other sources are regression test suites. The suites usually contain components with correct results for previous versions of the software. If some of the functionality in the new version overlaps the old version, the appropriate oracle information can be extracted. A working trusted program can serve as its own oracle in a situation where it is being ported to a new environment. In this case its intended behaviour should not change in the new environment.

Test Bed

A test bed is an environment that contains all the hardware and software needed to test a software component or a software system.

This includes the entire testing environment, for example, simulators, emulators, memory checkers, hardware probes, software tools, and all other items needed to support execution of the tests.

Software Quality

Two concise definitions for quality are found in the IEEE Standard Glossary
of Software Engineering Terminology:

1. Quality relates to the degree to which a system, system component, or process meets specified requirements.
2. Quality relates to the degree to which a system, system component, or process meets customer or user needs, or expectations.

In order to determine whether a system, system component, or process is of high quality we use what are called quality attributes. These are characteristics that reflect quality. For software artifacts we can measure the degree to which they possess a given quality attribute with quality metrics.

A **metric** is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute [2]. There are product and process metrics. A very commonly used example of a software product metric is software size, usually measured in lines of code (LOC). Two examples of commonly used process metrics are costs and time required for a given task. Many other examples are found in Grady [6]. Appendix I gives additional references that discuss metrics in depth. Quality metrics are a special kind of metric.

A **quality metric** is a quantitative measurement of the degree to which an item possesses a given quality attribute

Many different quality attributes have been described for software, for example, in IEEE Standards for Software Quality Metrics Methodology and work by Schulmeyer and Grady

Some examples of quality attributes with brief explanations are the following:

correctness—the degree to which the system performs its intended function reliability—the degree to which the software is expected to perform its required functions under stated conditions for a stated period of time

usability—relates to the degree of effort needed to learn, operate, prepare input, and interpret output of the software

integrity—relates to the system's ability to withstand both intentional and accidental attacks

portability—relates to the ability of the software to be transferred from one environment to another

maintainability—the effort needed to make changes in the software

interoperability—the effort needed to link or couple one system to another. Another quality attribute that should be mentioned here is testability. This attribute is of more interest to developers/testers than to clients.

It can be expressed in the following two ways:

1. the amount of effort needed to test the software to ensure it performs according to specified requirements (relates to number of test cases needed),
2. the ability of the software to reveal defects under testing conditions (some software is designed in such a way that defects are well hidden during ordinary testing conditions). Testers must work with analysts, designers and, developers throughout the software life system to ensure that testability issues are addressed.

Software Quality Assurance Group

The software quality assurance (SQA) group in an organization has ties to quality issues. The group serves as the customers' representative and advocate. Their responsibility is to look after the customers' interests. The software quality assurance (SQA) group is a team of people with the necessary training and skills to ensure that all necessary actions are taken during the development process so that the resulting software conforms to established technical requirements.

They work with project managers and testers to develop quality-related policies and quality assurance plans for each project. The group is also involved in measurement collection and analysis, record keeping, and reporting. The SQA team members participate in reviews, and audits (special types of reviews that focus on adherence to standards, guidelines, and procedures), record and track problems, and verify that corrections have been made. They also play a role in software configuration management.

Reviews

In contrast to dynamic execution-based testing techniques that can be used to detect defects and evaluate software quality, reviews are a type of static testing technique that can be used to evaluate the quality of a software artifact such as a requirements document, a test plan, a design document, a code component. Reviews are also a tool that can be applied to revealing defects in these types of documents. A definition follows.

A review is a group meeting whose purpose is to evaluate a software artifact or a set of software artifacts.

The composition of a review group may consist of managers, clients, developers, testers and other personnel depending on the type of artifact under review. A special type of review called an audit is usually conducted by a Software Quality Assurance group for the purpose of assessing compliance with specifications, and/or standards, and/or contractual agreements.

Software Testing Principles

Principles play an important role in all engineering disciplines and are usually introduced as part of an educational background in each branch of engineering. Figure 1.1 shows the role of basic principles in various engineering disciplines. Testing principles are important to test specialists/ engineers because they provide the foundation for developing testing knowledge and acquiring testing skills. They also provide guidance for defining testing activities as performed in the practice of a test specialist.

A principle can be defined as:

1. a general or fundamental, law, doctrine, or assumption;
2. a rule or code of conduct;
3. the laws or facts of nature underlying the working of an artificial device.

Extending these three definitions to the software engineering domain we can say that software engineering principles refer to laws, rules, or doctrines that relate to software systems, how to build them, and how they behave. In the software domain, principles may also refer to rules or codes of conduct relating to professionals who design, develop, test, and maintain software systems. Testing as a component of the software engineering discipline also has a specific set of principles that serve as guidelines for the tester. They guide testers in defining how to test software systems, and provide rules of conduct for testers as professionals. Glenford Myers has outlined such a set of execution-based testing principles in his pioneering book, The Art of Software Testing [9]. Some of these principles are described below. Principles 1–8, and 11 are derived directly from Myers' original set. The author has reworded these principles, and also has made modifications to the original set to reflect the evolution of testing from an art, to a quality-related process within the context of an engineering discipline. Note that the principles as stated below only relate to execution-based testing. Principles relating to reviews, proof of correctness, and certification as testing activities are not covered.

Principle 1. Testing is the process of exercising a software component using a selected set of test cases, with the intent of (i) revealing defects, and (ii) evaluating quality.
Software engineers have made great progress in developing methods to prevent and eliminate defects. However, defects do occur, and they have a negative impact on software quality. Testers need to detect these defects before the software becomes operational. This principle supports testing as an execution-based activity to detect defects. It also supports the separation of testing from debugging since the intent of the latter is to locate defects and repair the software. The term "software component" is used in this context to represent any unit of software ranging in size and complexity
from an individual procedure or method, to an entire software system. The term "defects" as used in this and in subsequent principles represents any deviations in the software that have a negative impact on its functionality, performance, reliability, security, and/or any other of its specified quality attributes. Bertolino, in the Guide to the Software Engineering Body of Knowledge, gives a view of testing as a ''dynamic process that executes a program on valued inputs''. This view, as well as the definition of testing given in Chapter 1, suggest that in addition to detecting defects, testing is also a process used to evaluate software quality. The purpose of the former has been described in the previous paragraph. In the case of the latter, the tester executes the software using test cases to evaluate properties such as reliability, usability, maintainability, and level of performance. Test results are used to compare the actual properties of the software to those specified in the requirements document as quality goals. Deviations or failure to achieve quality goals must be addressed. The reader should keep in mind that testing can have a broader scope as described in test process improvement models such as the TMM and other quality models. Reviews and other static analysis techniques are included under the umbrella of testing in the models. These techniques, and how they relate to detecting defects and evaluating quality will be described in subsequent chapters of this text.

Principle 2. When the test objective is to detect defects, then a good test case is one that has a high probability of revealing a yet undetected defect(s).

Principle 2 supports careful test design and provides a criterion with which to evaluate test case design and the effectiveness of the testing effort when the objective is to detect defects. It requires the tester to consider the goal for each test case, that is, which specific type of defect is to be detected by the test case. In this way the tester approaches testing in the same way a scientist approaches an experiment. In the case of the scientist there is a hypothesis involved that he/she wants to prove or disprove by means of the experiment. In the case of the tester, the hypothesis is related to the suspected occurrence of specific types of defects. The goal for the test is to prove/disprove the hypothesis, that is, determine if the specific defect is present/absent. Based on the hypothesis, test inputs are selected, correct outputs are determined, and the test is run. Results are analyzed to prove/disprove the hypothesis. The reader should realize that many resources are invested in a test, resources for designing the test cases, running the tests, and recording and analyzing results. A tester can justify the expenditure of the resources by careful test design so that principle 2 is supported.

Principle 3. Test results should be inspected meticulously.

Testers need to carefully inspect and interpret test results. Several erroneous and costly scenarios may occur if care is not taken.

For example:

- A failure may be overlooked, and the test may be granted a “pass” status when in reality the software has failed the test. Testing may continue based on erroneous test results. The defect may be revealed at some later stage of testing, but in that case it may be more costly and difficult to locate and repair.
- A failure may be suspected when in reality none exists. In this case the test may be granted a “fail” status. Much time and effort may be spent on trying to find the defect that does not exist. A careful re-examination of the test results could finally indicate that no failure has occurred.
- The outcome of a quality test may be misunderstood, resulting in unnecessary rework, or oversight of a critical problem.

Principle 4. A test case must contain the expected output or result.

It is often obvious to the novice tester that test inputs must be part of a test case. However, the test case is of no value unless there is an explicit statement of the expected outputs or results, for example, a specific variable value must be observed or a certain panel button that must light up.

Expected outputs allow the tester to determine (i) whether a defect has been revealed, and (ii) pass/fail status for the test. It is very important to have a correct statement of the output so that needless time is not spent due to misconceptions about the outcome of a test. The specification of test inputs and outputs should be part of test design activities. In the case of testing for quality evaluation, it is useful for quality goals to be expressed in quantitative terms in the requirements document if possible, so that testers are able to compare actual software attributes as determined by the tests with what was specified.

Principle 5. Test cases should be developed for both valid and invalid input conditions.

A tester must not assume that the software under test will always be provided with valid inputs. Inputs may be incorrect for several reasons.

For example, software users may have misunderstandings, or lack information about the nature of the inputs. They often make typographical errors even when complete/correct information is available. Devices may also provide invalid inputs due to erroneous conditions and malfunctions. Use of test cases that are based on invalid inputs is very useful for revealing defects since they may exercise the code in unexpected ways and identify unexpected software behaviour. Invalid inputs also help developers and testers evaluate the robustness of the software, that is, its ability to recover when unexpected events occur (in this case an erroneous input).

Principle 5 supports the need for the independent test group called for in Principle 7 for the following reason. The developer of a software component may be biased in the selection of test inputs for the component and specify only valid inputs in the test cases to demonstrate that the software works correctly. An independent tester is more apt to select invalid inputs as well.

Principle 6. The probability of the existence of additional defects in a software component is proportional to the number of defects already detected in that component.

What this principle says is that the higher the number of defects already detected in a component, the more likely it is to have additional defects when it undergoes further testing. For example, if there are two components A and B, and testers have found 20 defects in A and 3 defects in B, then the probability of the existence of additional defects in A is higher than B. This empirical observation may be due to several causes. Defects often occur in clusters and often in code that has a high degree of complexity
and is poorly designed. In the case of such components developers and testers need to decide whether to disregard the current version of the component and work on a redesign, or plan to expend additional testing resources on this component to insure it meets its requirements. This issue is especially important for components that implement mission or safety critical functions.

Principle 7. Testing should be carried out by a group that is independent of the development group.

This principle holds true for psychological as well as practical reasons. It is difficult for a developer to admit or conceive that software he/she has created and developed can be faulty. Testers must realize that (i) developers have a great deal of pride in their work, and (ii) on a practical level it may be difficult for them to conceptualize where defects could be found.

Even when tests fail, developers often have difficulty in locating the defects since their mental model of the code may overshadow their view of code as it exists in actuality. They may also have misconceptions or misunderstandings concerning the requirements and specifications relating to the software. The requirement for an independent testing group can be interpreted by an organization in several ways. The testing group could be implemented as a completely separate functional entity in the organization. Alternatively, testers could be members of a Software Quality Assurance
Group, or even be a specialized part of the development group, but in the latter case especially, they need the capability to be objective. Reporting to management that is separate from development can support their objectivity and independence. As a member of any of these groups, the principal duties and training of the testers should lie in testing rather than in development.
Finally, independence of the testing group does not call for an adversarial relationship between developers and testers. The testers should not play "gotcha" games with developers. The groups need to cooperate so that software of the highest quality is released to the customer.

Principle 8. Tests must be repeatable and reusable.

Principle 2 calls for a tester to view his/her work as similar to that of an experimental scientist. Principle 8 calls for experiments in the testing domain to require recording of the exact conditions of the test, any special events that occurred, equipment used, and a careful accounting of the results. This information is invaluable to the developers when the code is returned for debugging so that they can duplicate test conditions. It is also useful for tests that need to be repeated after defect repair. The repetition and reuse of tests is also necessary during regression test (the retesting of software that has been modified) in the case of a new release of the software. Scientists expect experiments to be repeatable by others, and testers should expect the same!

Principle 9. Testing should be planned.

Test plans should be developed for each level of testing, and objectives for each level should be described in the associated plan. The objectives should be stated as quantitatively as possible. Plans, with their precisely specified objectives, are necessary to ensure that adequate time and resources are allocated for testing tasks, and that testing can be monitored and managed. Test planning activities should be carried out throughout the software life cycle (Principle 10). Test planning must be coordinated with project planning. The test manager and project manager must work together to coordinate activities. Testers cannot plan to test a component on a given date unless the developers have it available on that date. Test risks must be evaluated.

For example, how probable are delays in delivery of software components, which components are likely to be complex and difficult to test, do the testers need extra training with new tools? A test plan template must be available to the test manager to guide development of the plan according to organizational policies and standards. Careful test planning avoids wasteful “throwaway” tests and unproductive and unplanned “test–patch–retest” cycles that often lead to poor-quality software and the inability to deliver software on time and within budget.

Principle 10. Testing activities should be integrated into the software life cycle.

It is no longer feasible to postpone testing activities until after the code has been written. Test planning activities as supported by Principle 10, should be integrated into the software life cycle starting as early as in the requirements analysis phase, and continue on throughout the software life cycle in parallel with development activities. In addition to test planning, some other types of testing activities such as usability testing can also be carried out early in the life cycle by using prototypes. These activities can continue on until the software is delivered to the users. Organizations can use process models like the V-model or any others that support the integration of test activities into the software life cycle [11].

Principle 11. Testing is a creative and challenging task [12].

Difficulties and challenges for the tester include the following:

- A tester needs to have comprehensive knowledge of the software engineering discipline.
- A tester needs to have knowledge from both experience and education as to how software is specified, designed, and developed.
- A tester needs to be able to manage many details.
- A tester needs to have knowledge of fault types and where faults of a certain type might occur in code constructs.
- A tester needs to reason like a scientist and propose hypotheses that relate to presence of specific types of defects.
- A tester needs to have a good grasp of the problem domain of the software that he/she is testing. Familiarly with a domain may come from educational, training, and work-related experiences.
- A tester needs to create and document test cases. To design the test cases the tester must select inputs often from a very wide domain.

Those selected should have the highest probability of revealing a defect (Principle 2). Familiarly with the domain is essential.

- A tester needs to design and record test procedures for running the tests.
- A tester needs to plan for testing and allocate the proper resources.
- A tester needs to execute the tests and is responsible for recording results.
- A tester needs to analyse test results and decide on success or failure for a test. This involves understanding and keeping track of an enormous amount of detailed information. A tester may also be required to collect and analyse test-related measurements.
- A tester needs to learn to use tools and keep abreast of the newest test tool advances.
- A tester needs to work and cooperate with requirements engineers, designers, and developers, and often must establish a working relationship with clients and users.
- A tester needs to be educated and trained in this specialized area and often will be required to update his/her knowledge on a regular basis due to changing technologies.

The Tester's Role in a Software Development Organization

Testing is sometimes erroneously viewed as a destructive activity. The tester's job is to reveal defects, find weak points, inconsistent behaviour, and circumstances where the software does not work as expected. As a tester you need to be comfortable with this role. Given the nature of the tester's tasks, you can see that it is difficult for developers to effectively test their own code (Principles 3 and 8). Developers view their own code as their creation, their "baby," and they think that nothing could possibly be wrong with it! This is not to say that testers and developers are adversaries. In fact, to be most effective as a tester requires extensive programming experience in order to understand how code is constructed, and where, and what kind of, defects are likely to occur. Your goal as a tester is to work with the developers to produce high-quality software that meets the customers' requirements. Teams of testers and developers are very common in industry, and projects should have an appropriate developer/tester ratio. The ratio will vary depending on available resources, type of project, and TMM level. For example, an embedded Realtime system needs to have a lower developer/tester ratio (for example, 2/1) than a simple data base application (4/1 may be suitable). At higher TMM levels where there is a well-defined testing group, the developer/ tester ratio would tend to be on the lower end (for example 2/1 versus 4/1) because of the availability of tester resources. Even in this case, the nature of the project and project scheduling issues would impact on the ratio.

In addition to cooperating with code developers, testers also need to work alongside with requirements engineers to ensure that requirements are testable, and to plan for system and acceptance test (clients are also involved in the latter). Testers also need to work with designers to plan for integration and unit test. In addition, test managers will need to cooperate with project managers in order to develop reasonable test plans,
and with upper management to provide input for the development and maintenance of organizational testing standards, polices, and goals. Finally, testers also need to cooperate with software quality assurance staff and software engineering process group members. In view of these requirements for multiple working relationships, communication and teamworking skills are necessary for a successful career as a tester. If you are employed by an organization that is assessed at TMM levels 1 or 2, you may find that there is no independent software test function in your organization. Testers in this case may be a part of the development group, but with special assignment to testing, or they may be part of the software quality assurance group. In fact, even at levels 3 and higher of the TMM the testers may not necessarily belong to a independent organizational entity, although that is the ideal case. However, testers should always have managerial independence from developers as described in Principle 8, and in the TMM at level 3. Testers are specialists, their main function is to plan, execute, record, and analyse tests. They do not debug software. When defects are detected during testing, software

should be returned to the developers who locate the defect and repair the code. The developers have a detailed understanding of the code, and are the best qualified staff to perform debugging. Finally, testers need the support of management. Developers, analysts, and marketing staff need to realize that testers add value to a software product in that they detect defects and evaluate quality as early as possible in the software life cycle. This ensures that developers release code with few or no defects, and that marketers can deliver software that satisfies the customers' requirements, and is reliable, usable, and correct. Low-defect software also has the benefit of reducing costs such as support calls, repairs to operational software, and ill will which may escalate into legal action due to customer dissatisfaction. In view of their essential role, testers need to have a positive view of their work. Management must support them in their efforts and recognize their contributions to the organization.

The Three-Step Process to Becoming a World-Class Testing Organization

The roadmap to become a world-class software testing organization is a simple three-step process, as follows:

1. Define or adopt a world-class software testing model.
2. Determine your organization's current level of software testing capabilities, competencies, and user satisfaction.
3. Develop and implement a plan to upgrade from your current capabilities, competencies, and user satisfaction to those in the world-class software testing model.

This three-step process requires you to compare your current capabilities, competencies, and user satisfaction against those of the world-class software testing model. This assessment will enable you to develop a baseline of your organization's performance. The plan that you develop will, over time, move that baseline from its current level of performance to a world-class level. Understanding the model for a world-class software testing organization and then comparing your organization will provide you with a plan for using the remainder of the material in this book.

Software testing is an integral part of the software-development process, which comprises the following four components (see Figure 1-1):

1. Plan (P): Devise a plan. Define your objective and determine the strategy and supporting methods to achieve it. You should base the plan on an assessment of your current situation, and the strategy should clearly focus on the strategic initiatives/key units that will drive your improvement plan.
2. Do (D): Execute the plan. Create the conditions and perform the necessary training to execute the plan. Make sure everyone thoroughly understands the objectives and the plan. Teach workers the procedures and skills they need to fulfil the plan and thoroughly understand the job. Then perform the work according to these procedures.
3. Check (C): Check the results. Check to determine whether work is progressing according to the plan and whether the expected results are being obtained. Check for performance of the set procedures, changes in conditions, or abnormalities that may appear. As often as possible, compare the results of the work with the objectives.
4. Act (A): Take the necessary action. If your check-up reveals that the work is not being performed according to the plan or that results are not what you anticipated, devise measures to take appropriate actions.

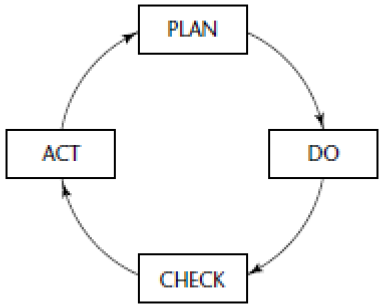


Fig. 1.1. The four components of the software-development process.

Testing involves only the “check” component of the plan-do-check-act (PDCA) cycle. The software development team is responsible for the three remaining components. The development team plans the project and builds the software (the “do” component); the testers check to determine that the software meets the needs of the customers and users. If it does not, the testers report defects to the development team. It is the development team that makes the determination as to whether the uncovered defects are to be corrected.

The role of testing is to fulfil the check responsibilities assigned to the testers; it is not to determine whether software can be placed into production. That is the responsibility of the customers, users, and development team.

Step 1: Define a World-Class Software Testing Model

There is no generally accepted model for a world-class software testing organization. However, analysing the best testing organizations among the more than 1,000 IT organizations affiliated with the Quality Assurance Institute (QAI) enabled QAI to identify the attributes of the best software testing organizations (see Figure 1-2). Organizations that follow this model report more effective and efficient testing than those that do not.

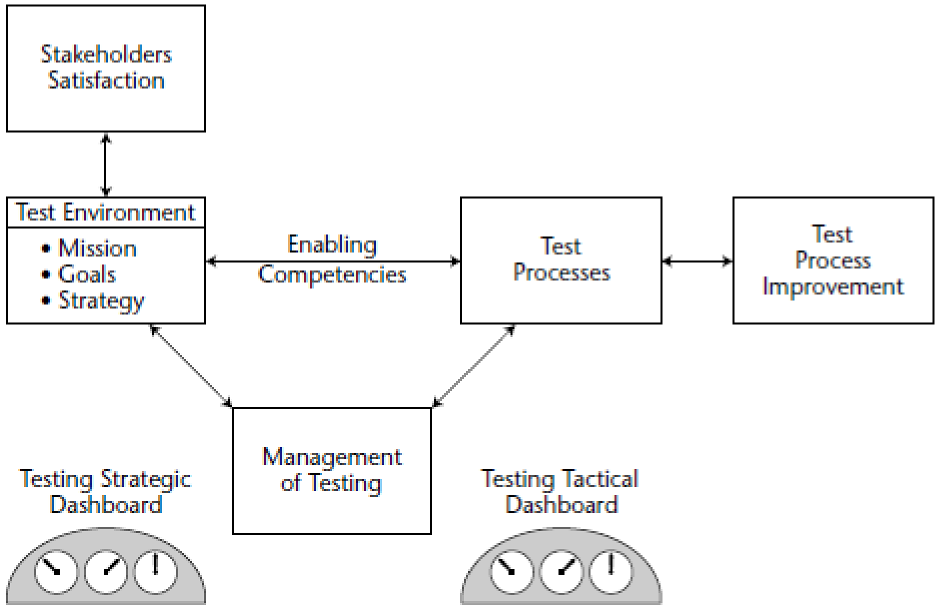


Figure 1-2 Model of a world-class software testing organization.

The world-class software testing model includes

Test environment. The conditions that management has put into place that both enable and constrain how testing is performed. The test environment includes management support, resources, work processes, tools, motivation, and so forth.

Process to test a single software project. The standards and procedures testers use to test.

Tester competency. The skill sets needed to test software in a test environment.

The three self-assessments that follow are for the above three attributes of a world-class software testing organization. The three self-assessments in this chapter correspond to the preceding three attributes of a world-class software testing organization. The world-class model of a software testing organization focuses on stakeholder satisfaction. This assumes a greater role for a world-class software testing organization than just testing against documented software requirements. Chapter 2 defines the many roles that software testing can adopt; however, those roles include much more than testing documented software requirements. They include testing for quality factors such as ease of use, meeting testing schedules and budgets, and minimizing the risks involved with any software project. According to the world-class model, the following parties have a vested interest in software testing:

Software customer. The party or department that contracts for the software to be developed.

Software user. The individual or group that will use the software once it is placed into production. (Note: This may be the customer or it may be parties other than the customer.)

Software developer. The individual or group that receives the requirements from the software user or assists in writing them, designing, building, and maintaining the software, as needed.

Development tester. The individual or group that performs the test function within the software development group.

IT management. The individual or group with responsibility for fulfilling the information technology mission. Testing supports fulfilling that mission.

Senior management. The CEO of the organization and other senior executives who are responsible for fulfilling the organization mission. Information technology is an activity that supports fulfilling that mission.

Auditor. The individual or group responsible for evaluating the effectiveness, efficiency, and adequacy of controls in the information technology area. Testing is considered a control by the audit function.

Project manager. The individual responsible for managing the building, maintaining, and/or implementing of software.

The test mission, strategy, and environment must be focused on stakeholder satisfaction. The mission defines the testing objectives; the strategy defines how the mission will be accomplished; and the environment provides the culture, processes, and tools that are conducive to effective and efficient software testing.

The test processes are those step-by-step procedures that the testers will follow to accomplish their assigned tasks. Test processes executed by trained and competent testers enable those testers to accomplish the defined test mission. The test processes need to be improved continually for two reasons: to make them more effective and efficient to use, and to incorporate updated approaches into testing new technologies and software development methodologies.

The responsibility for ensuring that the execution of the test processes meets the defined test mission lies with management. Management must ensure that testers are following and can accomplish the test plan, and that the plan will, in fact, accomplish the test objectives. If not, management should modify the plan to meet those objectives.

Management and testers need tools to enable them to fulfil their responsibilities. Two very important tools are the testing strategic dashboard and the testing tactical dashboard. The *testing strategic dashboard* includes key indicators such as user satisfaction, staff competency, and the percent of tests completed. The *testing tactical dashboard* includes test indicators such as the number of requirements tested and percent correct, defects uncovered, defects corrected and uncorrected, and the schedule and budget status.

Management must ensure that if you meet the testing tactical key indicators, you will, in fact, meet the objectives defined by the strategic key indicators. Customizing the World-Class Model for Your Organization You can customize the world-class model for software testing by defining the attributes of each of its components (refer to Figure 1-2). The material in this book explains the attributes of all the components: stakeholder satisfaction, test mission, test management and enabling competencies are discussed in Part 2. The test processes are explained in Parts 3 and 4. Test process improvement is described in Part 5 of this book.

As you read those parts of the book, you can customize those attributes based on the mission of your organization. For example, in describing a tester's competency, skill sets for testing COTS software and outsourced software will be listed. However, if your organization does not use COTS software or does not outsource the development of software, you would not need those skills in your testing staff. Likewise, if your testers are not responsible for testing security, you would not need a test processes for testing security.

The three self-assessments included in this chapter are based on the model in Figure 1-2. However, it is recognized that few testing organizations need all these testing capabilities and competencies. Therefore, you need to develop the model that is suited to your test mission.

Step 2: Develop Baselines for Your Organization

This section presents the following three self-assessment categories to enable you to compare your testing organization against the world-class model:

1. **Assessing the test environment.** This includes user satisfaction, management support, environment, planning, tools, test processes, measurement, quality control, and training.
2. **Assessing the process for testing individual software projects.** This category of assessment will assess your testing process against the seven-step process for testing individual software projects presented in Part 3 of this book.
3. **Assessing the competencies of software testers.** This self-assessment will be based on the 2006 Common Body of Knowledge (CBOK) developed by the Certification Board of the Software Certifications Organization. Each of the recommended ten competencies for software tester will be assessed. A more detailed assessment to be used in individuals to compare their specific test competencies against the 2006 CBOK is provided in Chapter 5.

Assessment 1: Assessing the Test Environment

During the past 25 years, the Quality Assurance Institute (QAI) has been studying what makes software testing organizations successful. As a result, QAI has identified the following eight criteria:

Test environment planning
Management support
Use of test processes
Test tools
Quality control
Test measurement
User satisfaction
Test training
When these eight criteria are in place and working, the result is normally a world-class testing organization.

The assessment process developed by QAI has five items to address within each of the eight criteria. The more of those items that are in place and working, the more likely that criteria will contribute to world-class testing. Figure 1-3 shows a cause-effect diagram indicating the areas to address, called *drivers*, which results in a world-class testing organization.

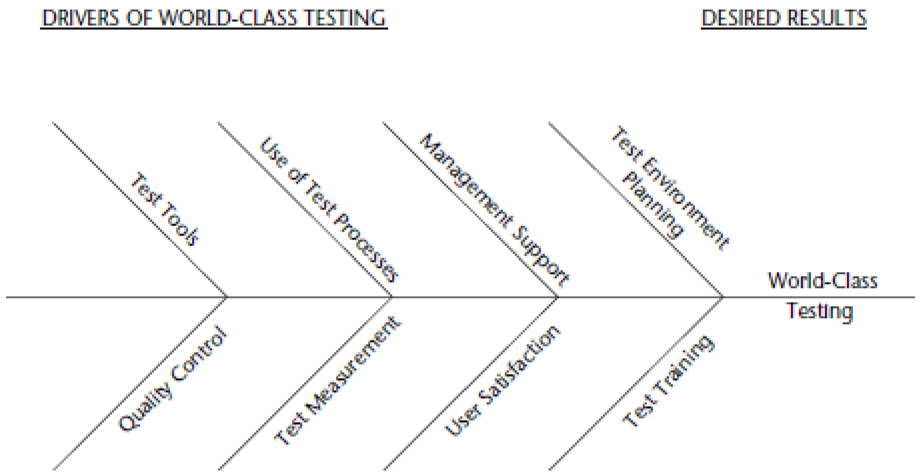


Figure 1-3 Overview of the testing environment.

Software testing organizations can use the results of this assessment in any one of the following three ways:

1. To determine their current testing environmental status versus the environment of a world-class testing organization. The responses to the items address will indicate an organization's strengths and weaknesses compared to the environment of a world-class testing organization.

2. To develop the goal/objectives to accomplish becoming a world-class testing organization. QAI's world-class criteria indicate a profile of the environment of a world-class testing organization. Achieving those objectives can lead you to become a more effective software testing organization.

3. To develop an improvement plan

By doing the assessment, you will develop a Footprint Chart that shows where improvement is needed. Those criteria in which you are deficient become the means for improving the environment of your software testing organization.

Implementation Procedures

This practice involves the following four tasks:

- Build the assessment team.
- Complete the assessment questionnaires.
- Build the footprint chart.
- Assess the results.

Building the Assessment Team

The assessment team should combine people who in totality possess the knowledge of how your organization manages software testing. Before the team is established, the areas to address should be reviewed to determine the makeup of the team. It is recommended that a matrix be prepared with the seven assessment criteria on one dimension and the recommended assessment team on the other. The matrix should indicate which assessment team member is knowledgeable about each of the seven assessment criteria. Once all seven criteria have been associated with an assessment team member, it can be concluded that the team is adequate to perform the assessment.

Completing the Assessment Questionnaire

The assessment questionnaire in Work Paper 1-1 consists of eight categories, with five items to address for each category. A Yes or No response should be made, as follows:

- A Yes response means all of the following:
- Criteria items are documented and in place.
- Criteria items are understood by testers.
- Criteria items are widely used, where applicable.
- Criteria items have produced some possible results.
- A No response means any of the following:
- No formal item in place.
- Criteria items are applied differently for different test situations.
- No consistency as to when used or used very seldom.
- No tangible results were produced.

The assessment team should read aloud each item and then discuss how that item is addressed in their testing environment. The results should be recorded on Work Paper 1-1. The assessment team may also wish to record comments that clarify the response and/or to provide insight in how that area may be improved.

Building the Footprint Chart

For this task, you should transcribe the results of Work Paper 1-1 onto Work Paper 1-2. To do so, total the number of Yes responses for each criterion. Then place a dot on Work Paper 1-2 on the line representing the number of Yes responses. For example, if you have three Yes responses for test training, you should place a dot on the test training line at the intersection of the line representing three Yes responses. Adot should be marked on the line representing all seven criteria for the number of Yes responses. Then connect the dots with a line, resulting in what is called a "footprint" of the status of your testing environment versus the environment of a world-class testing organization.

Assessing the Results

You should make the following two assessments regarding the footprint developed on the Work Paper 1-2:

1. **Assess the status of each criteria versus what that criteria should be in the world-class testing environment.** To do this, you need to look at the number of Yes responses you have recorded for each criterion versus a world-class organization, which would

have five Yes responses. For example, three Yes responses for test training would indicate that improvements could be made in your test training process. The two items that received No responses are indications of where improvements are needed to move your test training activities to a world-class level.

2. **Interpret your testing environment footprint chart.** The footprint in your Work Paper 1-2 provides an overview of your testing environment. Given the footprint, your assessment team should attempt to draw some conclusions about your testing environment. Three examples are given to help in drawing these conclusions, as shown in Figures 1-4, 1-5, and 1-6.

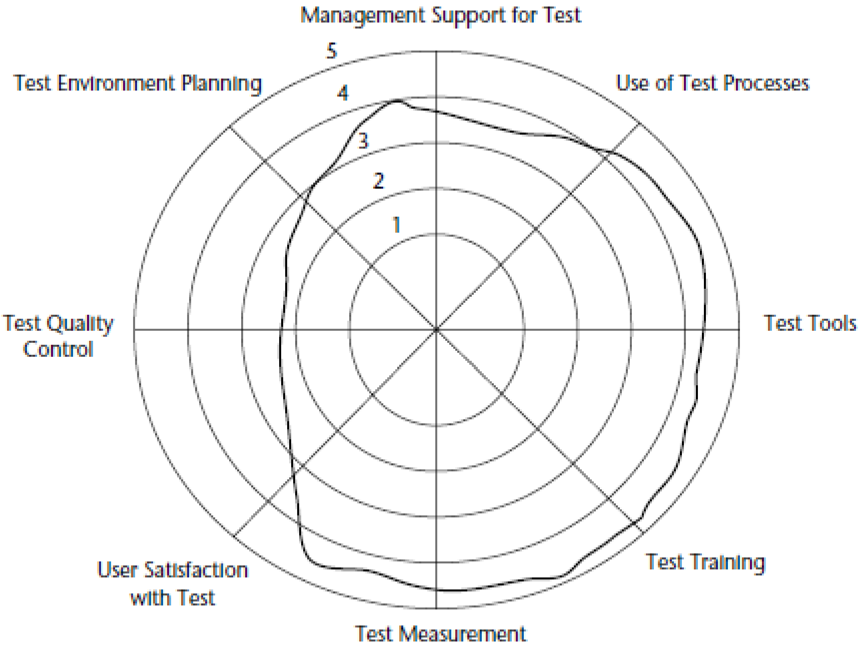


Figure 1-4 Example of a software testing organization using a test as a part of development.

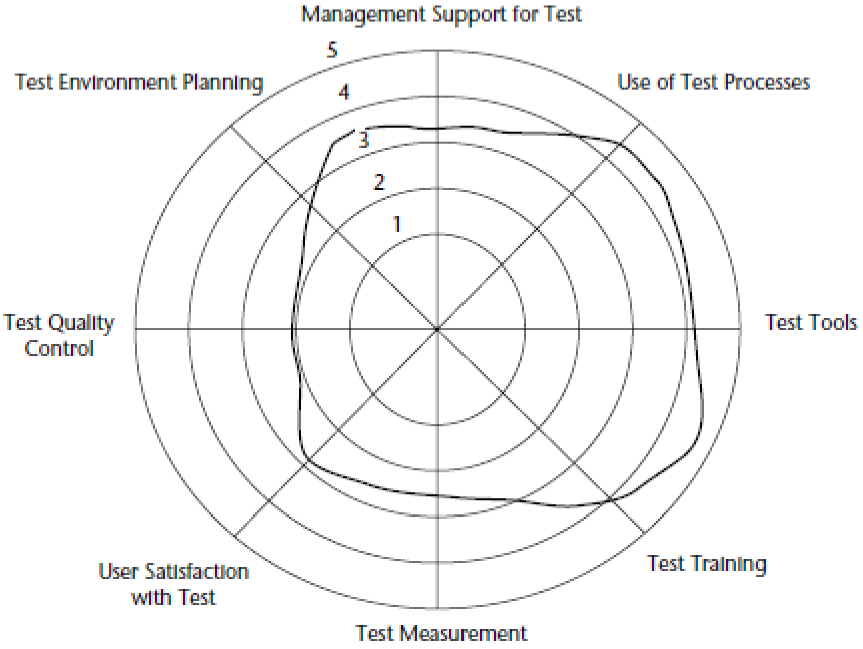


Figure 1-5 Example of a testing organization using, but not enforcing, the test process.

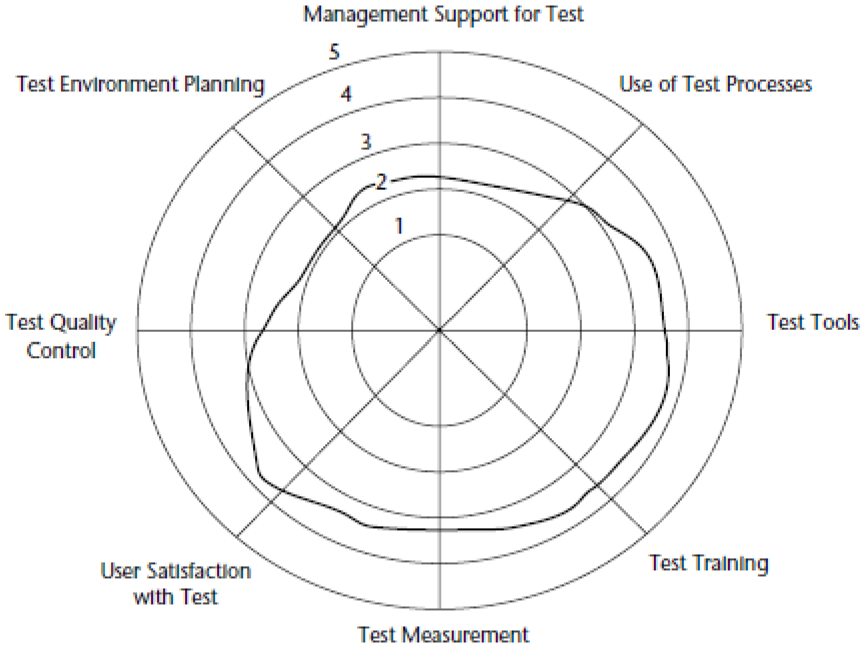


Figure 1-6 Example of a testing organization practicing testing as an art.

Verifying the Assessment

The following list of questions, if responded to positively, would indicate that the assessment has been performed correctly:

1. Does the assessment team comprise the knowledge needed to answer all of the items to address within the seven criteria?
2. Are the individual assessors free from any bias that would cause them not to provide proper responses to the items to address?
3. Was there general consensus among the assessment team to the response for each item to address?
4. Are the items to address appropriate for your testing organization?
5. Have the items to address been properly totaled and posted to the Footprint Chart Work Paper?
6. Does the assessment team believe the Footprint Chart is representative of your testing environment?
7. Does your assessment team believe that if they improve the items to address, which have No responses, the testing organization will become more effective?
8. Does your organization believe that the overall assessment made is representative of your environment?

Assessment 2: Assessing the Capabilities of Your Existing Test Processes

To assess the capabilities of your existing test processes, follow the same procedure that you used to assess your test environment. Note that you should use the same team for both assessments. The only change you will need is to substitute self-assessment questionnaires in assessing the test environment process with the self-assessment questionnaires for assessing the test processes included in this section.

The assessment of test processes will be divided into the following seven categories:

- Preparing for a software testing project
- Conducting test planning
- Executing the test plan
- Conducting acceptance testing
- Analyzing test results and preparing reports
- Testing the installation of software
- Post-test analysis

Note that these seven categories of test processes correspond to a seven-step software testing process presented in Part 3 of this book. Thus, each assessment will help you determine your strengths and weaknesses in each of the seven steps of the proposed software testing process. To conduct this self-assessment, answer the questionnaire in Work Paper 1-3 and post your results to Work Paper 1-4, as described in the preceding section.

Assessment 3: Assessing the Competency of Your Testers

This practice will enable you to assess your testing competencies against the ten skill categories in the Common Body of Knowledge (CBOK) for the Certified Software Tester (CSTE) certificate. At the conclusion of the assessment, you will develop a Foot print Chart that shows your competencies against the skill categories needed to become a CSTE. You can use the results to design a program for improving your personal test competencies.

Figure 1-7 shows a cause-effect diagram indicating the areas of competency assessment. In the diagram these are called the drivers that result in becoming a fully competent software tester. The drivers are, in fact, the ten CBOK skill categories.

Implementation Procedures

This practice involves performing the following four tasks:

1. Understand the CSTE CBOK.
2. Complete the assessment questionnaires.
3. Build the footprint chart.
4. Assess the results.

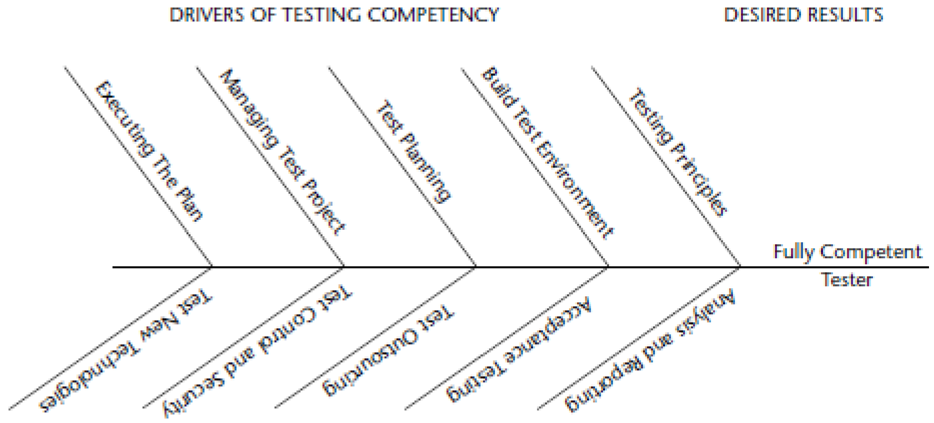


Figure 1-7 Test competency cause-effect diagram.

Understanding the CSTE CBOK

Before you can effectively evaluate your software test competencies, you need to understand the 2006 CSTE CBOK. The final version of the 2006 CSTE CBOK is available through the Software Certification Organization. The discussion draft version of the 2006 CSTE CBOK is included in Chapter 5 as a detailed skill-assessment questionnaire. This step requires you to read through the CBOK and to obtain clarifications of the material as necessary. The best source for these clarifications is the CSTE CBOK study guide, which is available from the Quality Assurance Institute (www.QAIworldwide.org).

Completing the Assessment Questionnaires

The assessment questionnaire in Work Paper 1-5 contains ten knowledge categories with 5 items in each category, for a total of 50 items to assess. For each item, a Yes or No response should be made. The meanings of the Yes and No responses are as follows:

- A Yes response means all of the following:
- You have had formal training, experience, or self-study supporting this skill item.
- You have actively used the skill in your personal or work life.
- You have accomplished some positive result using this skill item.
- A No response means any of the following:
- You do not understand the theory and concepts supporting the skill item.
- You have never used the skill item in a personal or work situation.
- You have used the skill item but you have never achieved any positive results.

Prior to answering each question, you should think through the meaning of the question. This may require referring back to the CSTE study guide. Using the Yes/No response criteria, you need to come to a consensus on whether a Yes/No response should be indicated for the skill item. The result of your assessment should be recorded on the appropriate questionnaire. You need to progress sequentially through the self-assessment questionnaires. Note that you may wish to make notes on the questionnaire to clarify your response or to indicate ideas on how you could improve your competency in that skill item.

Building the Footprint Chart

To build the footprint chart, transcribe the results of Work Paper 1-5 onto Work Paper 1-6. To do so, total the number of Yes responses for each of the ten knowledge categories. Then place a dot on Work Paper 1-6 on the lines corresponding to the knowledge category. For example, if you have three Yes responses for the Test Planning category, you should place a dot on the Test Planning line at the intersection of the line representing the three Yes responses. After you have placed all ten dots, draw a line to connect them. This line, called a *footprint*, represents the status of your testing competencies versus those specified in the CSTE CBOK.

Assessing the Results

You should make the following two assessments regarding the footprint you developed on Work Paper 1-6:

1. **Compare your results for each knowledge category versus what the knowledge category should be as indicated in the CSTE CBOK.** Any rating less than five Yes responses indicates a potential area of improvement in that knowledge category. An analysis of the CBOK knowledge categories will be helpful in determining where to focus improvement, as will studying the CSTE guide to identify areas for potential improvement.
2. **Compare your testing competencies against your current job responsibilities.** The footprint provides an overview of your current competencies. Using your current job description, develop another footprint, which you believe is needed to achieve your current job responsibilities. Any deficiencies should be your first objective for improvement; your second for improvement would be to achieve the skill competencies needed to become a CSTE.

Verifying the Assessment

A positive response to the following questions indicates that you have correctly performed the competency assessment: (Note: Any negative response to the following five questions would reduce the value in using this self-assessment to measure an individual tester's competency.)

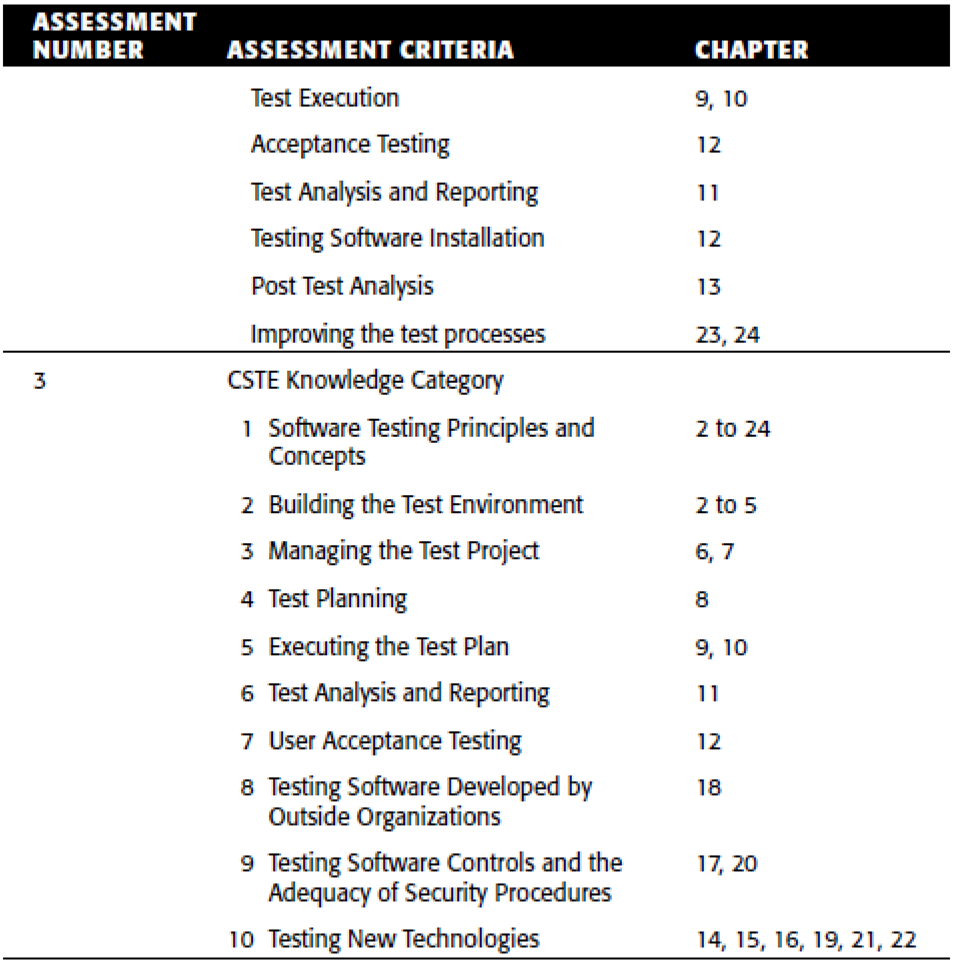
1. Do you have enough knowledge of the CSTE CBOK to understand the assessment questions?
2. Do you understand the skills required for each of the 50 assessment items in the questionnaires?
3. Do you understand the Yes and No response criteria, and have you used them in developing the competency assessment?
4. Do you believe the 50 assessment items fairly represent the competencies needed to be fully effective in software testing?
5. Do you believe that the 2006 CSTE CBOK used for this assessment is representative of your personal testing competencies?

Step 3: Develop an Improvement Plan

The objective of the action plan is to move software testing from where it is (the baseline) to where it should be (the goal). There is no one way to develop this plan. Some organizations want to implement the plan so it is on a "pay as you go basis." Other organizations are willing to invest in developing a significantly improved test process knowing that the payback will come after the process is developed and deployed.

The practices outlined in this book correspond to the three self-assessment footprints. If your organization is deficient in one or more components of the footprints, refer to the related chapter in this book that will help you develop your improvement plan, as shown in the following table:

ASSESSMENT NUMBER	ASSESSMENT CRITERIA	CHAPTER
1	Test environment assessment:	
	Test Environment Planning	2
	Management Support	2
	User Satisfaction	2
	Use of Process	3
	Test Tools	4
	Test Training	5
	Test Measurements	11
	Test Quality Control	2, 23, 24
2	Test Process Assessment:	
	Preparing for a Software testing Project	6
	Test Planning General	6, 7, 8
	Planning for specialized areas:	
	The Impact of Software Developmental Methodology When Testing	14
	Testing Client/Server Systems	15
	Testing Rapid Application Development	16
	Testing the Adequacy of Internal Control	17
	Testing Off-the-Shelf Software	18
	Testing in a Multi-Platform Environment	19
	Testing Security	20
	Testing a Data Warehouse	21
	Testing Web-based Systems	22



ASSESSMENT NUMBER	ASSESSMENT CRITERIA	CHAPTER
	Test Execution	9, 10
	Acceptance Testing	12
	Test Analysis and Reporting	11
	Testing Software Installation	12
	Post Test Analysis	13
	Improving the test processes	23, 24
3	CSTE Knowledge Category	
	1 Software Testing Principles and Concepts	2 to 24
	2 Building the Test Environment	2 to 5
	3 Managing the Test Project	6, 7
	4 Test Planning	8
	5 Executing the Test Plan	9, 10
	6 Test Analysis and Reporting	11
	7 User Acceptance Testing	12
	8 Testing Software Developed by Outside Organizations	18
	9 Testing Software Controls and the Adequacy of Security Procedures	17, 20
	10 Testing New Technologies	14, 15, 16, 19, 21, 22

ROLE OF TESTING

Testing plays an important role in achieving and assessing the quality of a software product. We improve the quality of the products as we repeat a *test–find defects–fix* cycle during development. We assess how good our system is when we perform system-level tests before releasing a product. Software testing is a verification process for software quality assessment and improvement. The activities for software quality assessment can be divided into two broad categories, namely, *static analysis* and *dynamic analysis*.

- **Static Analysis:** As the term “static” suggests, it is based on the examination of a number of documents, namely requirements documents, software models, design documents, and source code. Traditional static analysis includes code review, inspection, walk-through, algorithm analysis, and proof of correctness. It does not involve actual execution of the code under development. Instead, it examines code and reasons over all possible behaviours that might arise during run time. Compiler optimizations are standard static analysis.
- **Dynamic Analysis:** Dynamic analysis of a software system involves actual program execution in order to expose possible program failures. The behavioural and performance properties of the program are also observed. Programs are executed with both typical and carefully chosen input values. Often, the input set of a program can be impractically large. However, for practical considerations, a finite subset of the input set can be selected. Therefore, in testing, we observe some representative program behaviours and reach a conclusion about the quality of the system. Careful selection of a finite test set is crucial to reaching a reliable conclusion. By performing static and dynamic analyses, practitioners want to identify as many faults as possible so that those faults are fixed at an early stage of the software development.

Static analysis and dynamic analysis are complementary in nature, and for better effectiveness, both must be performed repeatedly and alternated. Practitioners and researchers need to remove the boundaries between static and dynamic analysis and create a hybrid analysis that combines the strengths of both approaches.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

School of Computing

Department of Computer Science and Engineering

UNIT - II

Data Communication and Computer Networks-SBS1302

UNIT II SIGNALS & ERRORS

Parallel and Serial Transmission - DTE/DCE/such as EIA-449, EIA-53O EIA-202 and x.21 interface - Interface standards - Modems - Guided Media Unguided Media - Performance - Types of Error - Error Detection - Error Corrections.

1. Parallel and Serial Transmission

Data transmission refers to the process of transferring data between two or more digital devices. Data is transmitted from one device to another in analog or digital format. Basically, data transmission enables devices or components within devices to speak to each other.Data is transferred in the form of bits between two or more digital devices. There are two methods used to transmit data between digital devices: serial transmission and parallel transmission. Serial data transmission sends data bits one after another over a single channel. Parallel data transmission sends multiple data bits at the same time over multiple channels.

Serial Transmission

When data is sent or received using serial data transmission, the data bits are organized in a specific order, since they can only be sent one after another. The order of the data bits is important as it dictates how the transmission is organized when it is received. It is viewed as a reliable data transmission method because a data bit is only sent if the previous data bit has already been received.

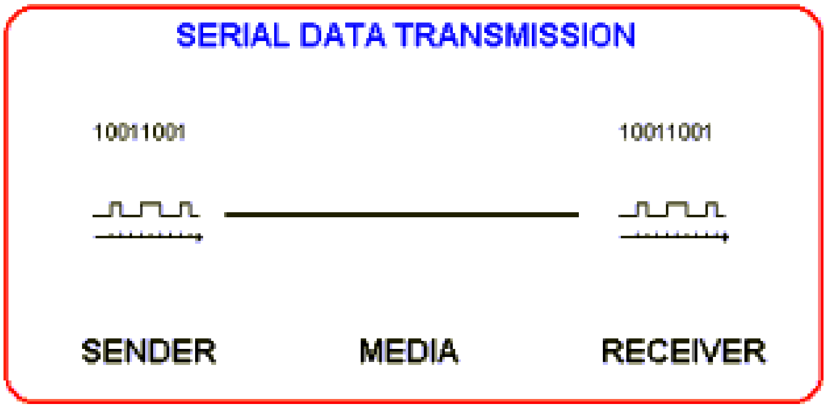


Fig 1 Serial Data Transmission

Serial transmission has two classifications: asynchronous and synchronous.

Asynchronous Serial Transmission

Data bits can be sent at any point in time. Stop bits and start bits are used between data bytes to synchronize the transmitter and receiver and to ensure that the data is transmitted correctly. The time between sending and receiving data bits is not constant, so gaps are used to provide time between transmissions.The advantage of using the asynchronous method is that no synchronization is required between the transmitter and receiver devices. It is also a more cost effective method. A disadvantage is that data transmission can be slower, but this is not always the case.

In asynchronous seria 0000000000000000l communication, the electrical interface is held in the **mark** position between characters. The start of transmission of a character is signaled by a drop in signal level to the **space** level. At this point, the receiver starts its clock. After one bit time (the start bit) come 8 bits of true data followed by one or more stop bits at the mark level.

The receiver tries to sample the signal in the middle of each bit time. The byte will be read correctly if the line is still in the intended state when the last stop bit is read.Thus the transmitter and receiver only have to have **approximately the same clock rate**. A little arithmetic will show that for a 10 bit sequence, the last bit will be interpreted correctly even if the sender and receiver clocks differ by as much as 5%. It is **relatively simple**, and therefore inexpensive. However, it has a **high overhead**, in that each byte carries at least two extra bits: a 20% loss of line bandwidth.

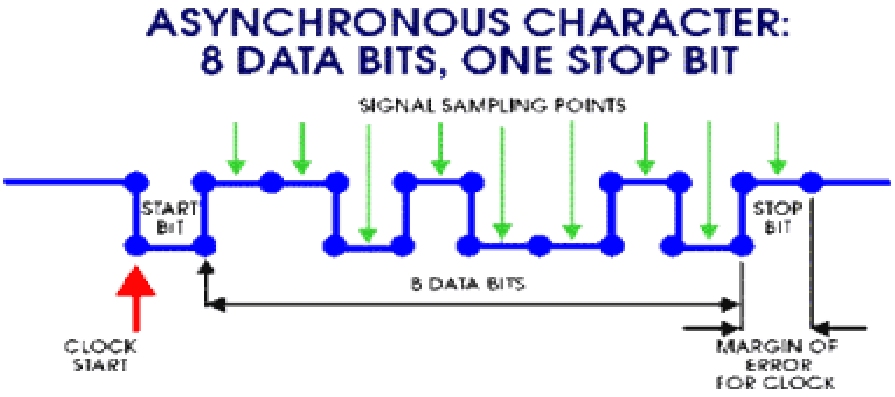


Fig 2 Asynchronous Transmission

Synchronous Serial Transmission

Data bits are transmitted as a continuous stream in time with a master clock. The data transmitter and receiver both operate using a synchronized clock frequency; therefore, start bits, stop bits, and gaps are not used. This means that data moves faster and timing errors are less frequent because the transmitter and receiver time is synced. However, data accuracy is highly dependent on timing being synced correctly between devices. In comparison with asynchronous serial transmission, this method is usually more expensive.Serial transmission is normally used for long-distance data transfer. It is also used in cases where the amount of data being sent is relatively small. It ensures that data integrity is maintained as it transmits the data bits in a specific order, one after another. In this way, data bits are received in-sync with one another.

The PS/2 mouse and keyboard implement a bidirectional synchronous serial protocol.

The bus is "**idle**" when both lines are high (open-collector). This is the only state where the keyboard/mouse is allowed begin transmitting data. The host has ultimate control over the bus and may inhibit communication at any time by pulling the Clock line low. The device (slave) always generates the clock signal. If the host wants to send data, it must first inhibit communication from the device by pulling Clock low. The host then pulls Data low and releases Clock. This is the "Request-to-Send" state and signals the device to start generating clock pulses.

Summary: Bus States

Data = high, Clock = high: *Idle state.*

Data = high, Clock = low: *Communication Inhibited.*

Data = low, Clock = high: *Host Request-to-Send*

Data is transmited 1 byte at a time:

- 1 start bit. This is always 0.
- 8 data bits, least significant bit first.
- 1 parity bit (odd parity - The number of 1's in the data bits plus the parity bit always add up to an odd number. This is used for error detection.).
- 1 stop bit. This is always 1.
- 1 acknowledge bit (host-to-device communication only)

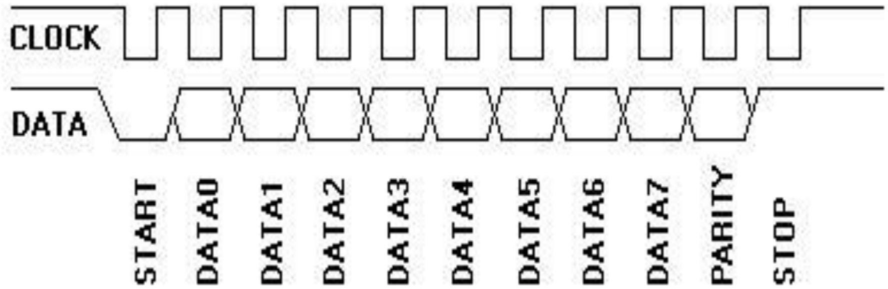


Fig 3 Synchronous Serial Transmission

Parallel transmission

When data is sent using parallel data transmission, multiple data bits are transmitted over multiple channels at the same time. This means that data can be sent much faster than using serial transmission methods.Parallel transmission (e.g. 8 bits). Each bit uses a separate wire to transfer data on a parallel link, a separate line is used as a clock signal. This serves to inform the receiver when data is available. In addition, another line may be used by the receiver to inform the sender that the data has been used, and its ready for the next data.

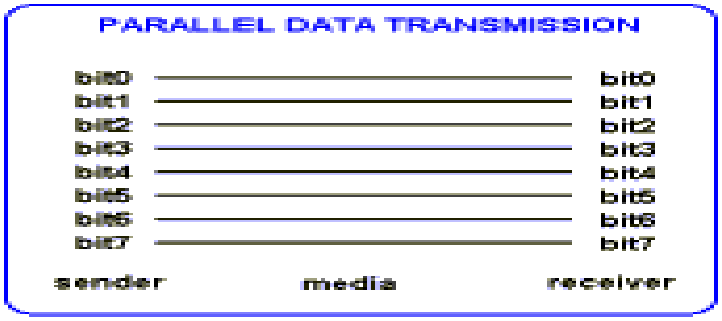


Fig 4 Parallel Transmission

Given that multiple bits are sent over multiple channels at the same time, the order in which a bit string is received can depend on various conditions, such as proximity to the data source, user location, and bandwidth availability. Two examples of parallel interfaces can be seen below. In the first parallel interface, the data is sent and received in the correct order. In the second parallel interface, the data is sent in the correct order, but some bits were received faster than others.

Advantages and Disadvantages of Using Parallel Data Transmission

The main advantages of parallel transmission over serial transmission are:

- it is easier to program;
- and data is sent faster.

Although parallel transmission can transfer data faster, it requires more transmission channels than serial transmission. This means that data bits can be out of sync, depending on transfer distance and how fast each bit loads. A simple of example of where this can be seen is with a voice over IP (VOIP) call when distortion or interference is noticeable. It can also be seen when there is skipping or interference on a video stream.

Parallel transmission is used when:

- a large amount of data is being sent;
- the data being sent is time-sensitive;
- and the data needs to be sent quickly.

A scenario where parallel transmission is used to send data is video streaming. When a video is streamed to a viewer, bits need to be received quickly to prevent a video pausing or buffering. Video streaming also requires the transmission of large volumes of data. The data being sent is also time-sensitive as slow data streams result in poor viewer experience.

2. Serial Interfaces

Serial links are simple, bidirectional links that require very few control signals. In a basic serial setup, data communications equipment (DCE) installed in a user's premises is responsible for establishing, maintaining, and terminating a connection. A modem is a typical DCE device.

A serial cable connects the DCE to a telephony network where, ultimately, a link is established with data terminal equipment (DTE). DTE is typically where a serial link terminates.

The distinction between DCE and DTE is important because it affects the cable pinouts on a serial cable. A DCE cable uses a female 9-pin or 25-pin connector, and a DTE cable uses a male 9-pin or 25-pin connector. To form a serial link, the cables are connected to each other. However, if the pins are identical, each side's transmit and receive lines are connected, which makes data transport impossible. To address this problem, each cable is connected to a null modem cable, which crosses the transmit and receive lines in the cable.

Serial Transmissions

In basic serial communications, nine signals are critical to the transmission. Each signal is associated with a pin in either the 9-pin or 25-pin connector. Table 1 lists and defines serial signals and their sources.When a serial connection is made, a serial line protocol—such as EIA-530, X.21, RS-422/449, RS-232, or V.35—begins controlling the transmission of signals across the line as follows:

The DCE transmits a DSR signal to the DTE, which responds with a DTR signal. After this handshake, the link is established and traffic can pass.

When the DTE device is ready to receive data, it sets its RTS signal to a marked state (all 1s) to indicate to the DCE that it can transmit data. (If the DTE is not able to receive data—because of buffer conditions, for example—it sets the RTS signal to all 0s.)

When the DCE device is ready to receive data, it sets its CTS signal to a marked state to indicate to the DTE that it can transmit data. (If the DCE is not able to receive data, it sets the CTS signal to all 0s.)

When the negotiation to send information has taken place, data is transmitted across the transmitted data (TD) and received data (RD) lines:

TD line—Line through which data from a DTE device is transmitted to a DCE device

RD line—Line through which data from a DCE device is transmitted to a DTE device

The name of the wire does not indicate the direction of data flow.

The DTR and DSR signals were originally designed to operate as a handshake mechanism. When a serial port is opened, the DTE device sets its DTR signal to a marked state. Similarly, the DCE sets its DSR signal to a marked state. However, because of the negotiation that takes place with the RTS and CTS signals, the DTR and DSR signals are not commonly used.The carrier detect and ring indicator signals are used to detect connections with remote modems. These signals are not commonly used.

Signal Polarity

Serial interfaces use a balanced (also called differential) protocol signaling technique. Two serial signals are associated with a circuit: the A signal and the B signal. The A signal is denoted with a plus sign (for example, DTR+), and the B signal is denoted with a minus sign (for example, DTR–). If DTR is low, then DTR+ is negative with respect to DTR–. If DTR is high, then DTR+ is positive with respect to DTR–.

By default, all signal polarities are positive, but sometimes they might be reversed. For example, signals might be mis wired as a result of reversed polarities.

Serial Clocking Modes

By default, a serial interface uses loop clocking to determine its timing source. For EIA-530 and V.35 interfaces, you can set each port independently to use one of the following clocking modes. X.21 interfaces can use only loop clocking mode. Loop clocking mode—Uses the DCE's receive (RX) clock to clock data from the DCE to the DTE.DCE clocking mode—Uses the transmit (TXC) clock, generated by the DCE specifically to be used by the DTE as the DTE's transmit clock.

Internal clocking mode—Uses an internally generated clock. The speed of this clock is configured locally. Internal clocking mode is also known as line timing. Both loop clocking mode and DCE clocking mode use external clocks generated by the DCE.

Serial Interface Clocking Modes

When an externally timed clocking mode (DCE or loop) is used, long cables might introduce a phase shift of the DTE-transmitted clock and data. At high speeds, this phase shift might cause errors. Inverting the transmit clock corrects the phase shift, thereby reducing error rates.

Serial Line Protocols

Serial interfaces support the following line protocols:

1. EIA-530
2. RS-232
3. RS-422/449
4. V.35
5. X.21

EIA-530

EIA-530 is an Electronic Industries Association (EIA) standard for the interconnection of DTE and DCE using serial binary data interchange with control information exchanged on separate control circuits. EIA-530 is also known as RS-530.The EIA-530 line protocol is a specification for a serial interface that uses a DB-25 connector and balanced equivalents of the RS-232 signals—also called V.24. The EIA-530 line protocol is equivalent to the RS-422 and RS-423 interfaces implemented on 25-pin connector.The EIA-530 line protocol supports both balanced and unbalanced modes. In unbalanced transmissions, voltages are transmitted over a single wire. Because only a single signal is transmitted, differences in ground potential can cause fluctuations in the measured voltage across the link. For example, if a 3-V signal is sent from one endpoint to another, and the receiving endpoint has a ground potential 1 V higher than the transmitter, the signal on the receiving end is measured as a 2-V signal.

Balanced transmissions use two wires instead of one. Rather than sending a single signal across the wire and having the receiving end measure the voltage, the transmitting device sends two separate signals across two separate wires. The receiving device measures the difference in voltage of the two signals (balanced sampling) and uses that calculation to evaluate the signal. Any differences in ground potential affect both wires equally, and the difference in the signals is still the same.

The EIA-530 interface supports asynchronous and synchronous transmissions at rates ranging from 20 Kbps to 2 Mbps.

RS-232

RS-232 is a Recommended Standard (RS) describing the most widely used type of serial communication. The RS-232 protocol is used for asynchronous data transfer as well as synchronous transfers using HDLC, Frame Relay, and X.25. RS-232 is also known as EIA-232.The RS-232 line protocol is very popular for low-speed data signals. RS-232 signals are carried as single voltages referred to a common ground signal. The voltage output level of these signals varies between –12 V and +12 V. Within this range, voltages between –3 V and +3 V are considered inoperative and are used to absorb line noise. Control signals are considered operative when the voltage ranges from +3 V to +25 V.

The RS-232 line protocol is an unbalanced protocol, because it uses only one wire and is susceptible to signal degradation. Degradation can be extremely disruptive, particularly when a difference in ground potential exists between the transmitting and receiving ends of a link.The RS-232 interface is implemented in a 25-pin D-shell connector and supports line rates up to 200 Kbps over lines shorter than 98 feet (30 meters).NOTERS-232 serial interfaces cannot function error-free with a clock rate greater than 200 KHz.

RS-422/449

RS-422 is a Recommended Standard (RS) describing the electrical characteristics of balanced voltage digital interface circuits that support higher bandwidths than traditional serial protocols like RS-232. RS-422 is also known as EIA-422.The RS-449 standard (also known as EIA-449) is compatible with RS-422 signal levels. The EIA created RS-449 to detail the DB-37 connector pinout and define a set of modem control signals for regulating flow control and line status.The RS-422/499 line protocol runs in balanced mode, allowing serial communications to extend over distances of up to 4,000 feet (1.2 km) and at very fast speeds of up to 10 Mbps.

Half-duplex transmission—In half-duplex transmission mode, transmissions occur in only one direction at a time. Each transmission requires a proper handshake before it is sent. This operation is typical of a balanced system in which two devices are connected by a single connection.Full-duplex transmission—In full duplex transmission mode, multiple transmissions can occur simultaneously so that devices can transmit and receive at the same time. This operation is essential when a single master in a point-to-multipoint system must communicate with multiple receivers.Multipoint transmission—RS-422/449 allows only a single master in a multipoint system. The master can communicate to all points in a multipoint system, and the other points must communicate with each other through the master.

V.35

V.35 is an ITU-T standard describing a synchronous, Physical Layer protocol used for communications between a network access device and a packet network. V.35 is most commonly used in the United States and Europe.The V.35 line protocol is a mixture of balanced (RS-422) and common ground (RS-232) signal interfaces. The V.35 control signals DTR, DSR, DCD, RTS, and CTS are single-wire common ground signals that are essentially identical to their RS-232 equivalents. Unbalanced signaling for these control signals is sufficient, because the control signals are mostly constant, varying at very low frequency, which makes single-wire transmission suitable. Higher frequency data and clock signals are sent over balanced wires.

X.21

X.21 is an ITU-T standard for serial communications over synchronous digital lines. The X.21 protocol is used primarily in Europe and Japan.The X.21 line protocol is a state-driven protocol that sets up a circuit-switched network using call setup. X.21 interfaces use a 15-pin connector with the following eight signals:

Signal ground (G)—Reference signal used to evaluate the logic states of the other signals. This signal can be connected to the protective earth (ground).

DTE common return (Ga)—Reference ground signal for the DCE interface. This signal is used only in unbalanced mode.

Transmit (T)—Binary signal that carries the data from the DTE to the DCE. This signal can be used for data transfer or in call-control phases such as Call Connect or Call Disconnect.

Receive (R)—Binary signal that carries the data from the DCE to the DTE. This signal can be used for data transfer or in call-control phases such as Call Connect or Call Disconnect.

Control (C)—DTE-controlled signal that controls the transmission on an X.21 link. This signal must be on during data transfer, and can be on or off during call-control phases.

Indication (I)—DCE-controlled signal that controls the transmission on an X.21 link. This signal must be on during data transfer, and can be on or off during call-control phases.

Signal Element Timing (S)—Clocking signal that is generated by the DCE. This signal specifies when sampling on the line must occur.

Byte Timing (B)—Binary signal that is on when data or call-control information is being sampled. When an 8-byte transmission is over, this signal switches to off.

Transmissions across an X.21 link require both the DCE and DTE devices to be in a ready state, indicated by an all 1s transmission on the T and R signals.

3. Modem

Modem is abbreviation for Modulator – Demodulator. Modems are used for data transfer from one computer network to another computer network through telephone lines. The computer network works in digital mode, while analog technology is used for carrying massages across phone lines.

Modulator converts information from digital mode to analog mode at the transmitting end and demodulator converts the same from analog to digital at receiving end. The process of converting analog signals of one computer network into digital signals of another computer network so they can be processed by a receiving computer is referred to as digitizing.

When an analog facility is used for data communication between two digital devices called Data Terminal Equipment (DTE), modems are used at each end. DTE can be a terminal or a computer.

DTE can be a terminal or a computer.

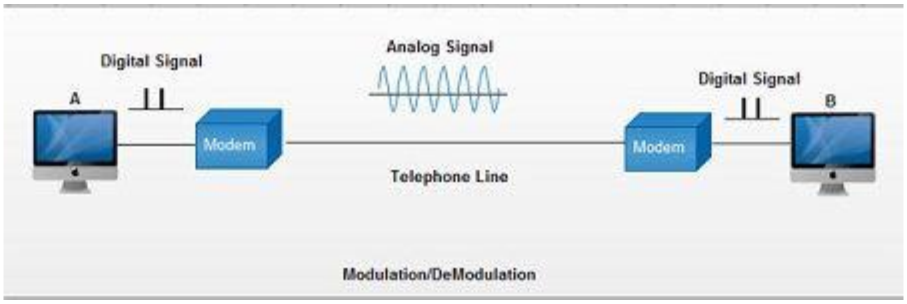
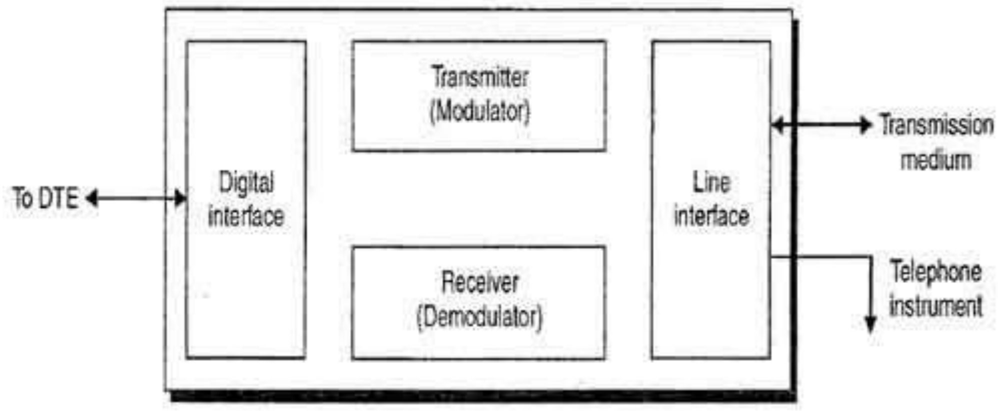


Fig 5 Modulation and Demodulation

The modem at the transmitting end converts the digital signal generated by DTE into an analog signal by modulating a carrier. This modem at the receiving end demodulates the carrier and hand over the demodulated digital signal to the DTE.



Building blocks of a modem

Fig 6 Building blocks of modem

The transmission medium between the two modems can be dedicated circuit or a switched telephone circuit. If a switched telephone circuit is used, then the modems are connected to the local telephone exchanges. Whenever data transmission is required connection between the modems is established through telephone exchanges.

Ready to Send

To begin with the Data Terminal Equipment or DTE (better known as a computer) sends a Ready To Send or RTS signal to the Data Communication Equipment or DCE (better known as a modem). This is sometimes known as a wakeup call and results in the modem sending a Data Carrier Detect or DCD signal to the receiving modem. There then follows a series of signals passed between the two until the communication channel has been established. This process is known as handshaking and helps to explain why, even now, some companies like CompuServe use the symbol of two hands grasping each other to mean being on-line. Of course, after that all it takes is for the second modem to send a Data Set Ready or DSR signal to its computer and wait for the Data Terminal Ready or DTR reply. When that happens the first modem sends a Clear To Send or CTS signal to the computer that started the whole process off and data can then be transmitted. It is as simple as that.

Types of Modems

- Modems can be of several types and they can be categorized in a number of ways.
- Categorization is usually based on the following basic modem features:

1. Directional capacity: half duplex modem and full duplex modem.

2. Connection to the line: 2-wire modem and 4-wire modem.

3. Transmission mode: asynchronous modem and synchronous modem.

Half duplex and full duplex Modems

Half duplex

1. A **half duplex modem** permits transmission in one direction at a time.

2. If a carrier is detected on the line by the modem, I gives an indication of the incoming carrier to the DTE through a control signal of its digital interface.

3. As long as they camel' IS being received; the modem does not give permission to the DTE to transmit data.

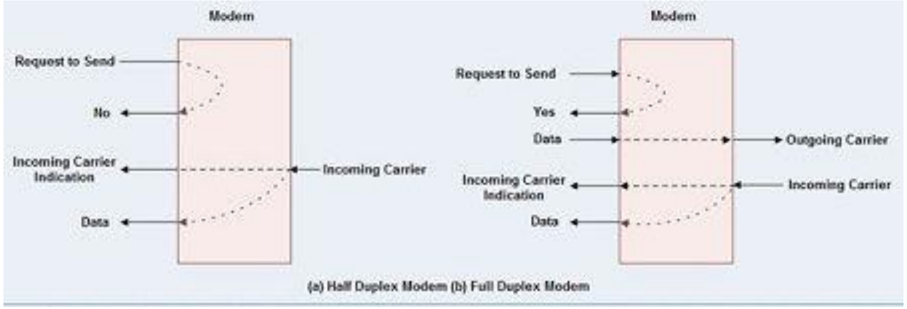


Fig 7 a) Half duplex modem b) Full duplex modem

Full duplex

• A **full duplex modem** allows simultaneous transmission in both directions.

• Therefore, there are two carriers on the line, one outgoing and the other incoming. **Wire and 4-wire Modems**

• The line interface of the modem can have a 2-wire or a 4-wire connection to transmission medium. 4-wire Modem

• In a 4-wire connection, one pair of wires is used for the outgoing carrier and the other pair is used for incoming carrier.

• Full duplex and half duplex modes of data transmission are possible on a 4- wire connection.

• As the physical transmission path for each direction is separate, the same carrier frequency can be used for both the directions.

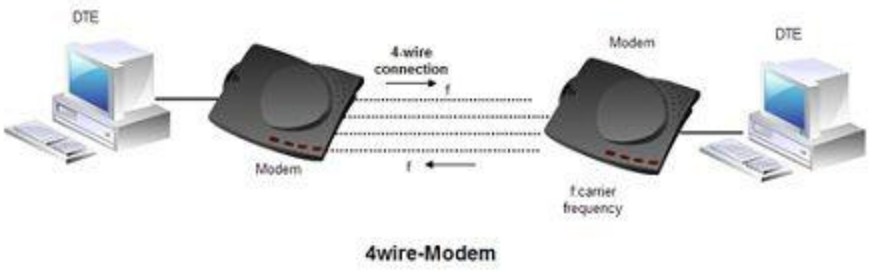


Fig 8 4-Wire modem

2-wire Modem

• 2-wire modems use the same pair of wires for outgoing and incoming carriers. A leased 2-wireconrlection is usually cheaper than a 4-wire connection as only one pair of wires is extended to the subscriber's premises. The data connection established through telephone exchange is also a 2-wire connection. In 2-wire modems, half duplex mode of transmission that uses the same frequency for the incoming and outgoing carriers can be easily implemented. For full duplex mode of operation, it is necessary to have two transmission channels, one for transmit direction and the other for receive direction. This is achieved by frequency division multiplexing of two different carrier frequencies. These carriers are placed within the bandwidth of the speech channel.

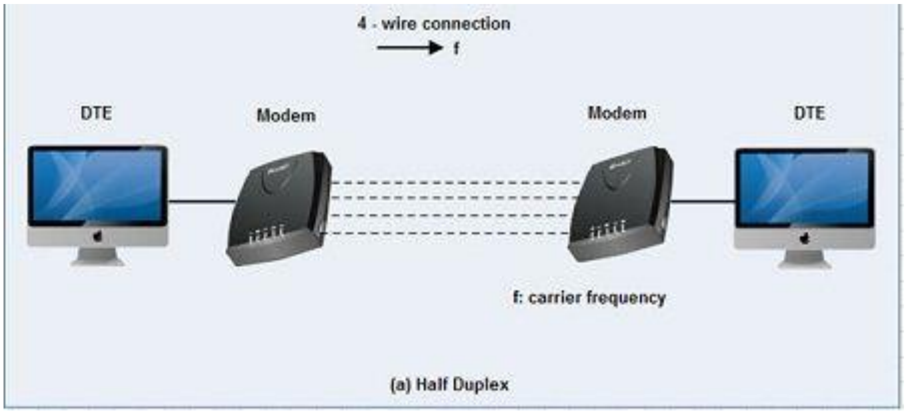


Fig 9 Half Duplex

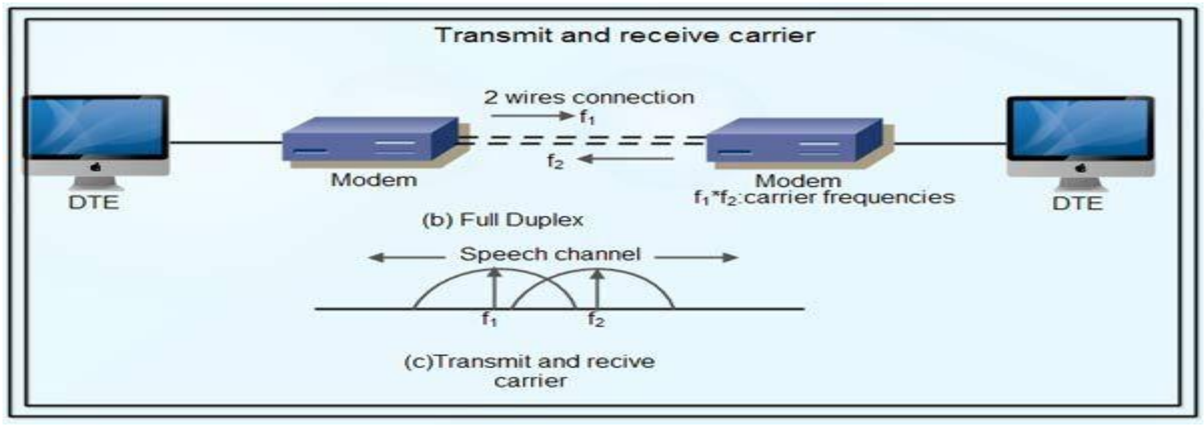

Transmit and receive carrier
2 wires connection
f_1
f_2
DTE
Modem
Modem
f_1*f_2:carrier frequencies
DTE
(b) Full Duplex
Speech channel
f_1
f_2
(c)Transmit and recive carrier

Fig 10 a) Full Duplex b) Transmit and receive carrier

Asynchronous & Synchronous Modems

Asynchronous Modem

Asynchronous modems can handle data bytes with start and stop bits. There is no separate timing signal or clock between the modem and the DTE. The internal timing pulses are synchronized repeatedly to the leading edge of the start pulse .

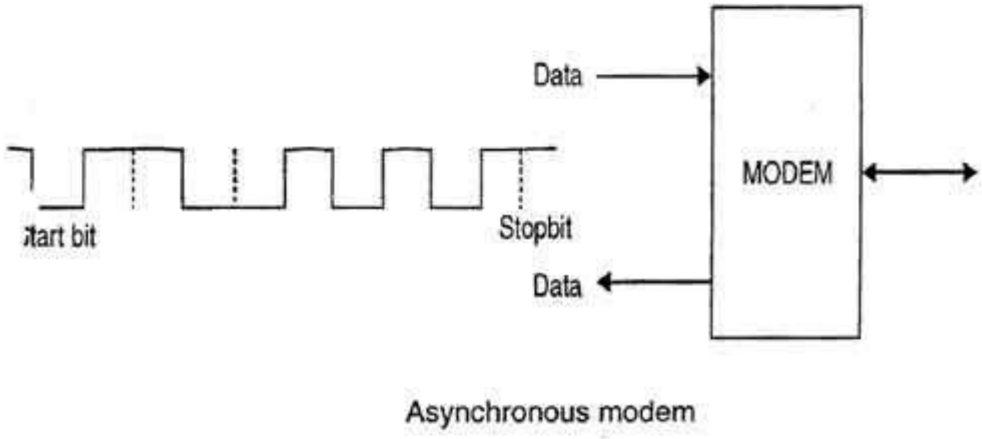


Fig 11 Asynchronous modem

Synchronous Modem

Synchronous modems can handle a continuous stream of data bits but requires a clock signal. The data bits are always synchronized to the clock signal. There are separate clocks for the data bits being transmitted and received. For synchronous transmission of data bits, the DTE can use its internal clock and supply the same to the modem.

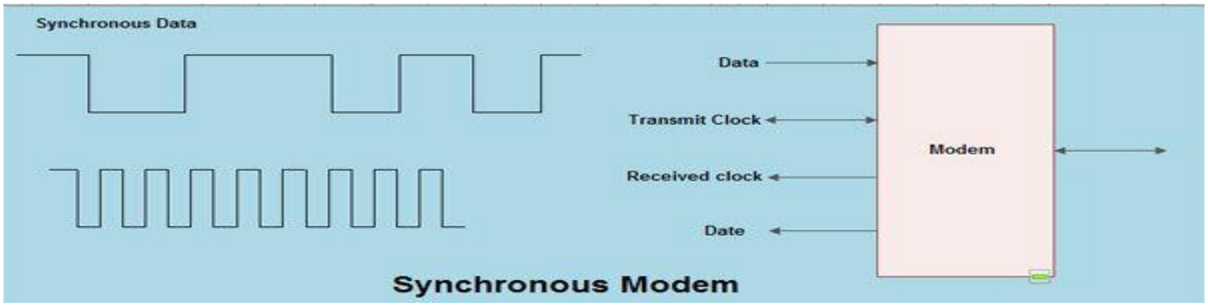


Fig 12 Synchronous Modem

Modulation techniques used for Modem:

The basic modulation techniques used by a modem to convert digital data to analog signals are :

• Amplitude shift keying (ASK).

• Frequency shift keying (FSK).

• Phase shift keying (PSK).

• Differential PSK (DPSK).

These techniques are known as the binary continuous wave (CW) modulation.

• Modems are always used in pairs. Any system whether simplex, half duplex or full duplex requires a modem at the transmitting as well as the receiving end.

• Thus a modem acts as the electronic bridge between two worlds - the world of purely digital signals and the established analog world.

4. Transmission Media

A **transmission medium** (plural *transmission media*) is a material substance (solid, liquid, gas, or plasma) which can propagate energy waves. For example, the transmission medium for sound received by the ears is usually air, but solids and liquids may also act as transmission media for sound.

Transmission media are the physical pathways that connect computers, other devices, and people on a network. Each transmission medium requires specialized network hardware that is compatible with that medium, and most networks need to use combination of transmission media types selected based on the network's needs and prevailing conditions. The term **transmission medium** can also refer to the technical device which employs the material substance to transmit or guide the waves. Thus an optical fiber or a copper cable can be referred to as a transmission medium. The absence of a material medium (the vacuum of empty space) can also be thought of as a transmission medium for electromagnetic waves such as light and radio waves. While material substance is not required for electromagnetic waves to propagate, such waves are usually affected by the transmission media through which they pass, for instance by absorption or by reflection or refraction at the interfaces between media.

A transmission medium can be classified as a:

- Linear medium, if different waves at any particular point in the medium can be superposed;
- Bounded medium, if it is finite in extent, otherwise unbounded medium;
- Uniform medium or homogeneous medium, if its physical properties are unchanged at different points;
- Isotropic medium, if its physical properties are the same in different directions.

Types of Transmission media:

The means through which data is transformed from one place to another is called transmission or communication media. There are two categories of transmission media used in computer communications.

- **GUIDEDMEDIA**
- **UNGUIDEDMEDIA**

5. Guided Media:

Bounded media are the physical links through which signals are confined to narrow path. These are also called guide media. Bounded media are made up oa external conductor (Usually Copper) bounded by jacket material. Bounded media are great for LABS because they offer high speed, good security and low cast. However, some time they cannot be used due distance communication. Three common types of bounded media are used of the data transmission. There are several types of cable which are commonly used with LANs. In some cases, a network will utilize only one type of cable, other networks will use a variety of cable types. The type of cable chosen for a network is related to the network's topology, protocol, and size. Understanding the characteristics of different types of cable and how they relate to other aspects of a network is necessary for the development of a successful network.

a. Unshielded Twisted Pair (UTP)Cable
b. Shielded Twisted Pair (STP)Cable
c. Coaxial Cable
d. Fiber Optic Cable

TWISTED PAIR CABLE :

e. The most popular network cabling is Twisted pair. It is light weight, easy to install, inexpensive and support many different types of network. It also supports the speed of **100 mps.** Twisted pair cabling is made of pairs of solid or stranded copper twisted along each other. The number of pairs in the cable depends on the type. The copper core is usually **22-AWG or 24-AWG, as** measured on the American wire gauge standard.

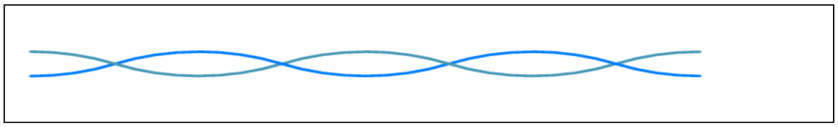


Fig 13 Twisted pair wire

There are two types of twisted pairs cabling

1. Unshielded twisted pair(UTP)

2. Shielded twisted pair(STP)

Unshielded Twisted Pair (UTP)Cable

Twisted pair cabling comes in two varieties: shielded and unshielded. Unshielded twisted pair (UTP) is the most popular and is generally the best option for school networks.

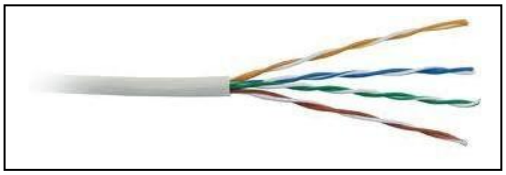


Fig 14 Unshielded Twisted Pair

The quality of UTP may vary from telephone-grade wire to extremely high-speed cable. The cable has four pairs of wires inside the jacket. Each pair is twisted with a different number of twists per inch to help eliminate interference from adjacent pairs and other electrical devices.

The tighter the twisting, the higher the supported transmission rate and the greater the cost per foot. The EIA/TIA (Electronic Industry Association / Telecommunication Industry Association) has established standards of UTP and rated five categories of wire.

Table 1 Categories of Unshielded Twisted Pair

Type	**Use**
Category 1	Voice Only (Telephone Wire)
Category 2	Data to 4 Mbps (LocalTalk)
Category 3	Data to 10 Mbps (Ethernet)
Category 4	Data to 20 Mbps (16 Mbps Token Ring)
Category 5	Data to 100 Mbps (Fast Ethernet)

CHARACTERISTICS_

- Easy to install
- High speed capacity
- High attenuation
- Effective to EMI
- 100 meter limit

ADVANTAGES :

- Easy installation
- Capable of high speed for LAN
- Low cost

DISADVANTAGES :

- Short distance due to attenuation

Unshielded Twisted Pair Connector Shielded twisted pair (STP)

It is similar to UTP but has a mesh shielding that's protects it from EMI which allows for higher transmission rate. IBM has defined category for STP cable.

Type 1: STP features two pairs of 22-AWG

Type 2: This type include type 1 with 4 telephone pairs

Type 3: This type feature two pairs of standard shielded 26-AWG

Type 4: This type of STP consists of 1 pair of standard shielded 26-AWG

Type 5: This type consist of shielded 26-AWG wire

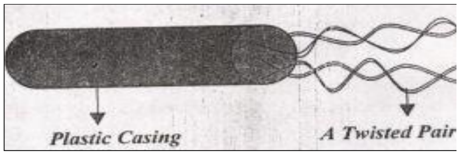


Fig 15 Twisted Pair

CHARACTERISTICS:

□ Medium cost

□ Easy to install

□ Higher capacity than UTP

□ Higher attenuation, but same as UTP

□ Medium immunity from EMI

□ 100 meter limit

ADVANTAGES :

□ Faster than UTP and coaxial &Shielded

DISADVANTAGES :

- More expensive than UTP and coaxial
- More difficult installation
- High attenuation rate

The standard connector for unshielded twisted pair cabling is an RJ-45 connector. This is a plastic connector that looks like a large telephone-style connector (See fig. 2.2). a slot allows the RJ-45 to be inserted only one way. RJ stands for Registered Jack, implying that the connector follows a standard borrowed from the telephone industry. This standard designates which wire goes with each pin inside the connector.

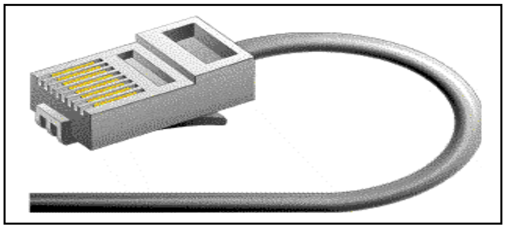


Fig 16 RJ-45

Shielded Twisted Pair (STP) Cable

A disadvantage of UTP is that it may be susceptible to radio and electrical frequency interference. Shielded twisted pair |(STP) is suitable for environments with electrical interference; however, the extra shielding can make the cables quite bulky. Shielded twisted pair is often used on networks using Token Ring topology.

Coaxial Cable

Coaxial cabling has a single copper conductor at its center. A plastic layer provides insulation between the center conductor and the braided metal shield (See fig. 3). The metal shield helps to block any outside interference from fluorescent lights, motors, and other computers.

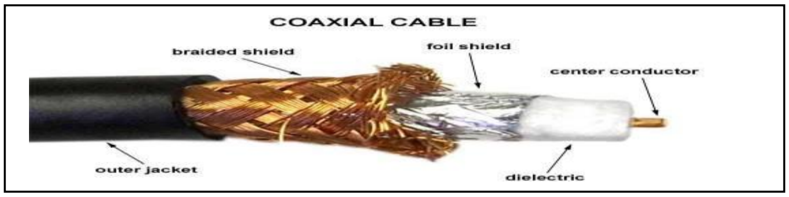


Fig 17 Coaxial Cable

Although coaxial cabling is difficult to install, it is highly resistant to signal interference. In addition, it can support grater cable lengths between network devices than twisted pair cable. The two types of coaxial cabling are thick coaxial and thin coaxial.

Thin coaxial cable is also referred to as thinnet. 10base2 refers to the specifications for thin coaxial cable carrying Ethernet signals. The 2 refers to the approximate maximum segment length being 200 meters. In actual fact the maximum segment length is 185 meters. Thin coaxial cable is popular in school networks, especially linear bus networks.

Thick coaxial cable is also referred to as thicknet. 10base refers to the specifications for thick coaxial cable carrying Ethernet signals. The 5 refers to the maximum segment length being 500 meters. Thick coaxial cable has an extra protective plastic cover that helps keep moisture away from the center conductor. This makes thick coaxial a great choice when running longer lengths in a linear bus network. One disadvantage of thick coaxial is that it does not bend easily and is difficult to install.

Here the most common coaxial standards

- 50-Ohm RG-7 or RG-11 : used with thick Ethernet.
- 50-Ohm RG-58 : used with thin Ethernet
- 75-Ohm RG-59 : used with cable television
- 93-Ohm RG-62 : used with ARCNET.

CHARACTERISTICS :

- Low cost
- Easy to install
- Up to 10Mbpscapacity
- Medium immunity form EMI
- Medium of attenuation

ADVANTAGES :

- Inexpensive
- Easy to wire
- Easy to expand
- Moderate level of EMI immunity

DISADVANTAGE :

- Single cable failure can take down an entire network

Coaxial Cable Connectors

The most common type of connector used with coaxial cables is the Bayone-Neill-Concelmnan (BNC) connector (See fig. 4). Different types of adapters are available for BNC connectors, including a T-connector, barrel connector, and a terminator. Connectors on the cable are the weakest points in any network. To help avoid problems with your network, always use the BNC connectors that crimp, rather than screw, onto the cable.

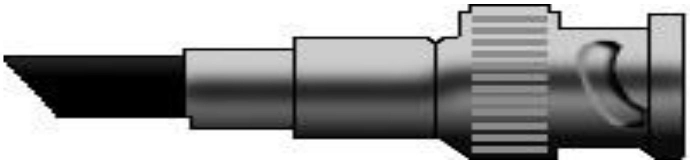


Fig 18 BNC Connector

Fiber Optic Cable

Fiber optic cabling consists of a center glass core surrounded by several layers of protective materials. It transmits light rather than electronic signals eliminating the problem of electrical interference.

This makes it ideal for certain environments that contain a large amount of electrical interference. It has also made it the standard for connecting networks between buildings, due to its immunity to the effects of moisture and lighting.

Fiber optic cable has the ability to transmit signals over much longer distances than coaxial and twisted pair. It also has made it the standard for connecting networks between buildings, due to its immunity to the effects of moisture and lighting.

Fiber optic cable has the ability to transmit signals over mush longer distances than coaxial and twisted pair. It also has the capability to carry information at vastly greater speeds. This capacity broadens communication possibilities to include services such as video conferencing and interactive services. The cost of fiber optic cabling is comparable to copper cabling; however it is more difficult to install and modify. 10BaseF refers to the specifications for fiber optic cable carrying Ethernet signals.

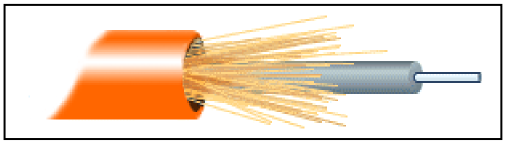


Fig 19 Fiber Optic Cable

Facts about fiber optic cables:

- □ Outer insulating jacket is made of Teflon or PVC.
- □ Kevlar fiber helps to strengthen the cable and prevent breakage.
- □ A plastic coating is used to cushion the fiber center.
- □ Center (core) is made of glass or plastic fibers.

CHARACTERISTICS:

- Expensive
- Very hard to install
- Capable of extremely high speed
- Extremely low attenuation
- No EMI interference

ADVANTAGES :

- Fast
- Low attenuation
- No EMI interference

DISADVANTAGES :

- Very costly
- Hard to install

Fiber Optic Connector

The most common connector used with fiber optic cable is an ST connector. It is barrel shaped, similar to a BNC connector. A newer connector, the SC, is becoming more popular. It has a squared face and is easier to connect in a confined space.

Table 2 Ethernet Cable Summary

Specification	Cable Type	Maximum length
10BaseT	Unshielded Twisted Pair	100 meters
10Base2	Thin Coaxial	185 meters
10Base5	Thick Coaxial	500 meters
10BaseF	Fiber Optic	2000 meters
100BaseT	Unshielded Twisted Pair	100 meters
100BaseTX	Unshielded Twisted Pair	220 meters

Installing Cable Guidelines

When running cable, it is best to follow a few simple rules:

- Always use more cable than you need. Leave plenty of slack.
- Test every part of a network as you install it. Even if it is brand new, it may have problems that will be difficult to isolate later.
- Stay at least 3 feet away from fluorescent light boxes and other sources of electrical interference.

- If it is necessary to run cable across the floor, cover the cable with cable protectors.
- Label both ends of each cable.
- Use cable ties (not tape) to keep cables in the same location together.

Where to put the cable?

The use of the surface allows cables to run along the outer edges of common floors covered by metal conduits that are attached at the room's floorboards.

Advantage:

This method is popular because it provides protection from electromagnetic interference.

Disadvantages:

- They are difficult to move or modify if a network expands or changes.
- Cable can also be installed under floors or over ceilings.
- Under floor cabling is usually housed in steel ducts or trenches.

Advantages:

- It is difficult to tap and is resistant to breaks and cuts.
- Over-ceiling cabling competes with air conditioning, lighting, and power conducts that some times squeeze out this popular, inexpensive emethod.

Who is responsible forcabling?

- Telephone installers
- LANi nstaller
- Network Manager

Often, a network manager must work with:

- LAN vendors
- Cabling vendors
- Cable installation firms

- □ Individual users
- □ Building owners or managers

As far as cabling is concerned, his / her job is to be responsible for:

- □ Installing
- □ Moving
- □ Changing
- □ Inventorying
- □ Regularly testing cables

6. UNBOUNDED MEDIA Or Unguided

Unguided transmission media are methods that allow the transmission of data without the use of physical means to define the path it takes. Unguided media provide a means for transmitting electromagnetic waves but do not guide them; examples are propagation through air, vacuum and seawater. For unguided media, the bandwidth of signal produced by the transmitting antenna is more important than the medium in determining transmission characteristics. One key property of signals transmitted by antenna is directionality. Unguided transmission media is data signals that flow through the air. They are not guided or bound to a channel to follow.

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication. Signals are normally broadcast through free space and thus are available to anyone who has a device receiving them. Unguided signals can travel from the source to destination in several ways: ground propagation, sky propagation, and line-of-sight propagation.

In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance. Ground waves have carrier frequencies up to 2 MHz. AM radio is an example of ground wave propagation.In sky propagation, higher frequency radio waves radiate upward into the ionosphere (the layer of atmosphere where the particles exist as ions) where they are reflected back to the earth. This

type of transmission allows for greater distances with lower output power.

It is sometimes called double hop propagation. It operates in the frequency range of 30 – 85

MHz. Because it depends on the earth's ionosphere, it changes with the weather and time of day. The signal bounces off of the ionosphere and back to the earth. Ham radios operate in this range. Other books called this Ionospheric propagation.

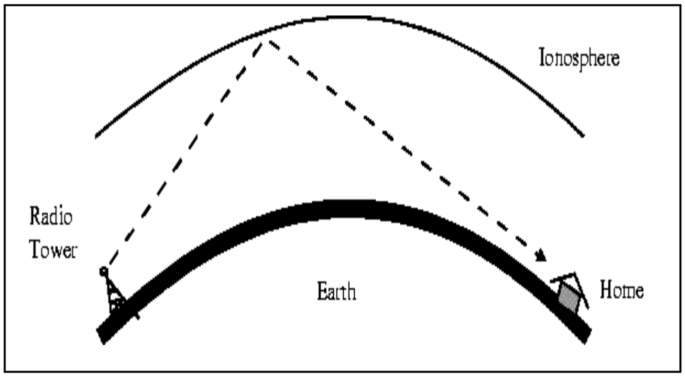


Fig 20 Ionospheric Propagation

In line-of-sight propagation, very high-frequency signals are transmitted in straight lines directly from antenna to antenna. Antennas must be directional, facing each other and either tall enough or close enough together not to be affected by the curvature the earth. Line-of-sight propagation is tricky because radio transmission cannot be completely focused.

It is sometimes called space waves or tropospheric propagation. It is limited by the curvature of the earth for ground-based stations (100 km, from horizon to horizon). Reflected waves can cause problems. Examples are: FM radio, microwave and satellite.

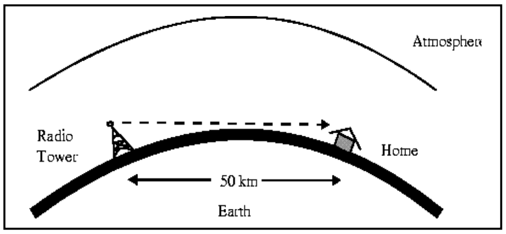


Fig 21 Line-of-sight Propagation

The section of the electromagnetic spectrum defined as radio waves and microwaves is divided into eight ranges, called bands, each regulated by government authorities. These bands are rated from very low frequency (VLF) to extremely high frequency (EHF).

We can divide wireless transmission into three broad groups: radio waves, microwaves, and infrared waves.

Examples of Unguided media are:

a. microwave
b. radio waves
c. infrared waves
d. Satellites

Radio Waves

Electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves. Radio waves are omni directional. When antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna.

The omni directional property has a disadvantage too. The radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band.

Radio waves, particularly those of low and medium frequencies, can penetrate walls. This characteristic can be both an advantage and disadvantage. It is an advantage because, for example, an AM radio can receive signals inside a building. It is a disadvantage because we cannot isolate a communication to just inside or outside a building.

Radio is the transmission of signals by modulation of electromagnetic waves with frequencies below those of visible light. Electromagnetic radiation travels by means of oscillating electromagnetic fields that pass through the air and the vacuum of space. Information is carried by systematically changing (modulating) some property of the radiated waves, such as amplitude, frequency, phase, or pulse width. When radio waves pass an electrical conductor, the oscillating fields induce an alternating current in the conductor. This can be detected and transformed into sound or other signals that carry information.

Characteristics:

- Directed Waves
- Noise Concurrency

- Radio Wave's Directness
- Unlimited Range
- Interference

ADVANTAGES :

- Can carry a message instantaneously over a wide area.
- Aerials to receive them are simpler than for microwaves.
- Wires are not needed as they travel through air, thus, a cheaper form of communication.

DISADVANTAGES:

- The range of frequencies that can be accessed by existing technology is limited, so there is a lot of competition amongst companies for the use of the frequencies.
- Travel in a straight line, so repeater stations may be needed.

Microwaves

Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves.

Microwaves are unidirectional. When an antenna transmits microwave waves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas. The following describes some characteristics of microwave propagation:

- Microwave propagation is line-of-sight. Since towers with the mounted antennas need to be in direct sight of each other. This also set a limit on the distance between stations depending on the local geography. Towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles does not allow two short towers to communicate by using microwaves. Typically the line of sight due to the Earth's curvature is only 50 km to the horizon. Repeaters are often needed for long-distance communication.
- Very high frequency microwaves cannot penetrate walls. This characteristic can be a

disadvantage if receivers are inside the buildings.

- The microwave band is relatively wide, almost 299 GHz. Therefore wider sub bands can be assigned, and a high data rate is possible.
- Use of certain portions of the band requires permission from authorities.

Advantages :

- No cables needed
- Multiple channels available
- Wide bandwidth

Disadvantages:

- Line-of-sight will be disrupted if any obstacle, such as new buildings, are in theway
- Signal absorption by the atmosphere. Microwaves suffer from attenuation due to atmospheric conditions.
- Towers are expensive to build

Infrared Waves

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 mm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. When we use our infrared remote control, we do not interfere with the use of the remote of our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

Infrared (IR) light is electromagnetic radiation with a wavelength between 0.7 and 300 micrometers, which equates to a frequency range between approximately 1 and 430 THz.

IR wavelengths are longer than that of visible light, but shorter than that of terahertz radiation microwaves. Bright sunlight provides an irradiance of just over 1 kilowatt per square meter at sea level. Of this energy, 527 watts is infrared radiation, 445 watts is visible light, and 32 watts is ultraviolet radiation.

Advantages :

- Many things are controlled by infrared.
- Sensors are invisible to the naked eye.
- They are very reliable.

Disadvantages:

- Most infrared sensors must be lined up or they will not work

Satellite

Satellites are transponders (units that receive on one frequency and retransmit on another) that are set in geostationary orbits directly over the equator. These geostationary orbits are 36, 000 km from the Earths's surface. At this point, the gravitational pull of the Earth and the centrifugal force of Earth's rotation are balanced and cancel each other out. Centrifugal force is the rotational force placed on the satellite that wants to fling it out into the space.

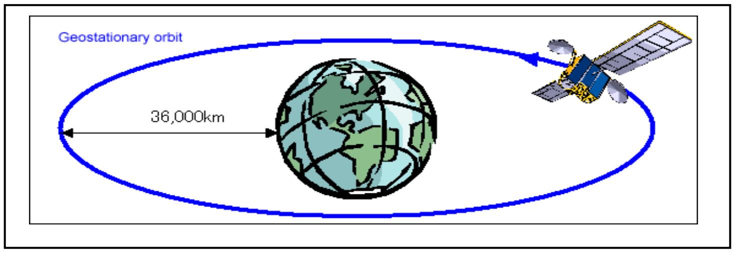


Fig 22 Satellite Communication

The uplink is the transmitter of data to the satellite. The downlink is the receiver of data. Uplinks and downlinks are also called Earth stations because they are located on the Earth. The footprint is the "shadow" that the satellite can transmit to, the shadow being the area that can receive the satellite's transmitted signal.

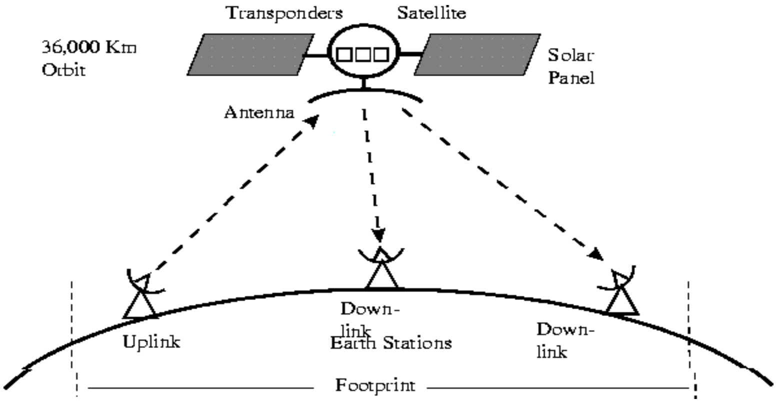


Fig 23 Uplink and Downlink

Advantages of satellite communication :

Availability

The biggest advantage of satellite Internet access is its availability compared to other Internet connection types. Satellite Internet access is a way for those who do not have access to terrestrial broadband connections such as cable or DSL to have access to high-speed Internet access. Satellite also is one of the only ways to receive Internet service in areas where telephone lines are not available.

Speed

Satellite Internet access is much faster than dial-up, with entry-level service tiers typically providing approximately 1 mbps download speeds--nearly 18 times faster than a dial- up modem. Faster speeds are generally available at higher service tiers. In general, the highest speeds available to home satellite Internet customers are slightly slower than the highest speeds offered by cable and DSL providers. Additionally, many satellite providers limit the amount of data that can be downloaded during short time periods to curb frequent large file transfers.

Latency

Satellite Internet connections are high-latency, meaning that a great deal of time is required for packets of information to travel to the satellite and back. The total delay can amount to about one second from the time that you send a request to the Internet to the time that a reply is received. Satellite Internet providers use various technologies to make this delay less noticeable to the end user and create an acceptable experience for browsing the Web. However, the latency makes a satellite Internet connection unsuitable for high-speed gaming.

Reliability

Home-based satellite Internet connections are generally no less reliable than terrestrial broadband. However, all satellite communication is subject to interruption during periods of heavy snow or rainfall. Talk to other customers about their experiences if you live in an area where either of these are common. The likelihood of weather-related interruptions is lessened with a larger satellite dish, which some providers offer.

Cost

The equipment costs several hundred dollars to purchase, and some types of installations incur additional fees. Additionally, the monthly cost for satellite Internet tends to be slightly higher than the cost of cable or DSL. There are ways of reducing the up-front cost. The equipment can be leased rather than purchased, and discounts or rebates may be available. Sometimes, installation fees are included in the lease price.

7. ERRORS, DETECTION & CORRECTION

Errors in the data are basically caused due to the various impairments that occur during the process of transmission.When there is an imperfect medium or environment exists in the transmission it prone to errors in the original data.

Transmission errors are usually detected at the physical layer of the OSI model. Transmission errors are usually corrected at the data link layer of the OSI model.

Types of Errors

Single-bit: In a single-bit error, only one bit in the data unit has changed.
Burst :A burst error means that two or more bits in the data unit have changed.

Single-bit error

The change in one bit in the whole data sequence , is called "Single bit error". Occurrence of single bit error is very rare in serial communication system. This type of error

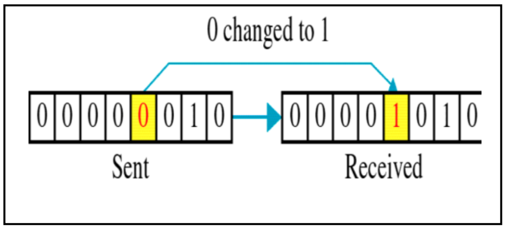


Fig 24 Single bit Error

ccurs only in parallel communication system, as data is transferred bit wise in single line, ere is chance that single line to be noisy.

Burst error

The change of set of bits in data sequence is called "Burst error". The burst error is calculated in from the first bit change to last bit change.

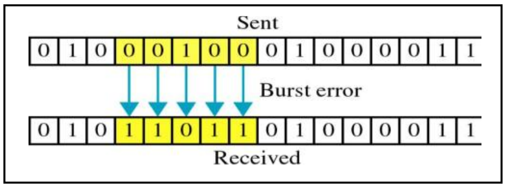


Fig 25 Burst Error

8. Error Detection

Redundancy-is one error detection mechanism. Redundancy is the concept of addition of extrabitstoamessageforerrordetection.Inordertodetectandcorrecttheerrorsinthedata communication we add some extra bits to the original data. These extra bits are nothing but the redundant bits which will be removed by the receiver after receiving the data.

Theirpresenceallowsthereceivertodetectorcorrectcorruptedbits.Insteadofrepeatingthe entire data stream, a short group of bits may be attached to the entire data stream. This technique is called redundancy because the extra bits are redundant to the information: they are discarded as soon as the accuracy of the transmission has been determined.

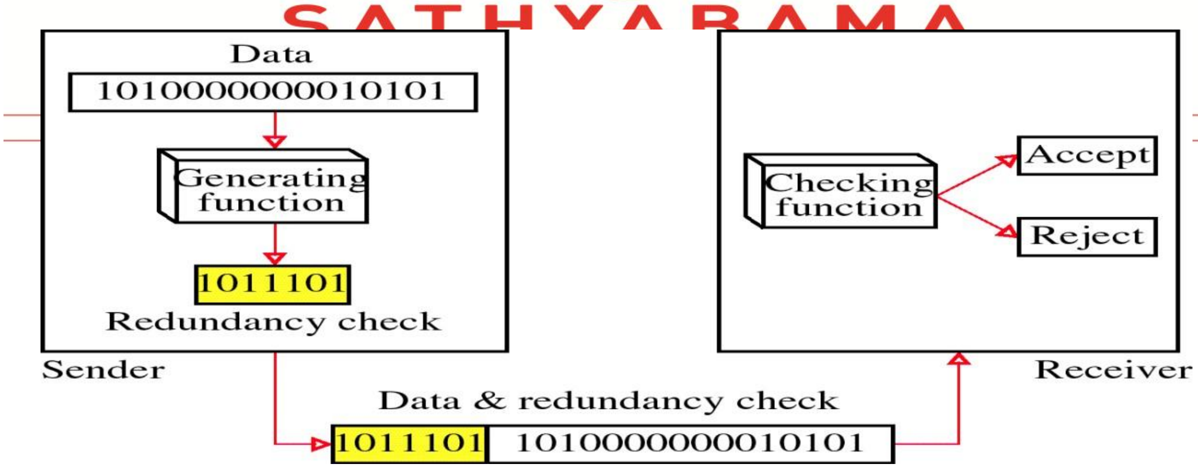


Fig 26 Redundancy

Four types of redundancy checks follows:

a) Vertical redundancy check(VRC)

b) Longitudinal redundancy check(LRC)

c) Cyclic redundancy check(CRC)

d)Checksum

Vertical redundancy check(VRC)

In this technique, a redundant bit ,called a parity bit is added to every data unit so that the total number of 1s becomes even.

Performance: VRC can detect only an odd numbers of errors; it cannot detect an even number of errors.

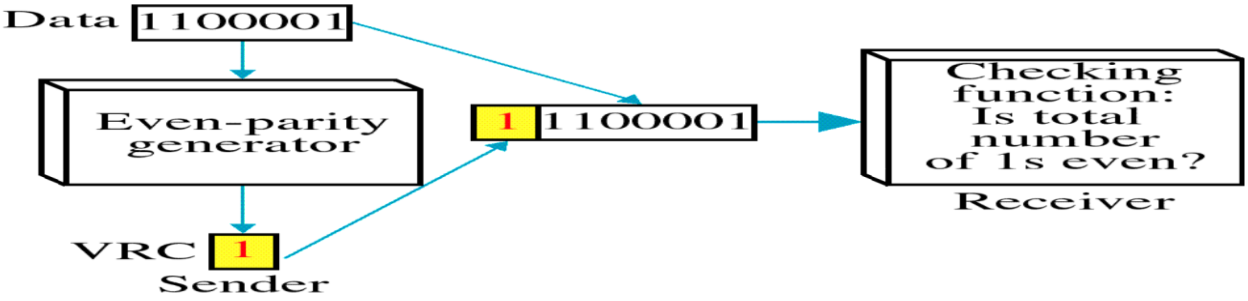


Fig 27 Virtual Redundancy Check

Example: Imagine the sender wants to send the word "world.” In ASCII the five characters are coded as

<--------1110111 1101111 1110010 11011001100100

w o r l d

Each of the first four characters has an even numbers of 1s,so the parity bit is a 0.The last character("d"),however, has three 1s(an odd number),so the parity bit is a 1 to make the total number of 1s even. The following shows the actual bits sent(the parity bits are underlined)

<-------11101110 11011110 11100100 1101100011001001

w o r l d

Example: Now suppose the word "world" in the previous example is received by the receiver but corrupted during transmission.

<------111*1*1110 11011110 1110*1*100 1101100011001001

w o r l d

The receiver counts the 1s in each character and comes up with even and odd numbers(7,6,5,4,4.)The receiver knows that the data are corrupted, discard them, and ask for retransmission.

Longitudinal redundancy check(LRC)

In LRC,a block of bits is organized in a table(rows and columns)

For example, instead of sending a block of 32 bits, we organize them in a table made of four rows and eight columns. We then calculate the parity bit for each column and create a new row of eight bits**.**

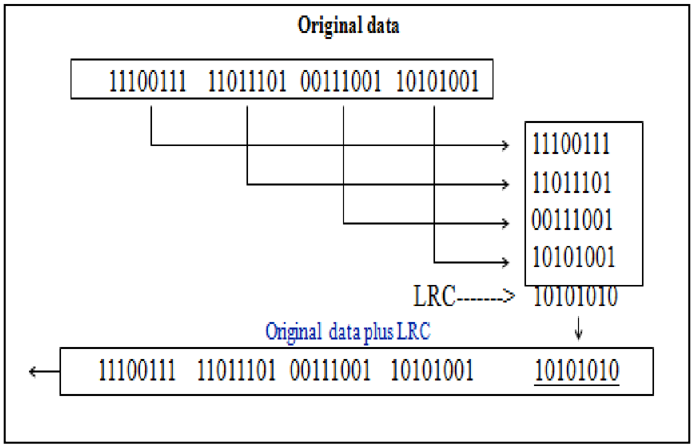


Fig 28 Longitudinal redundancy check

Example: Suppose the following block is sent

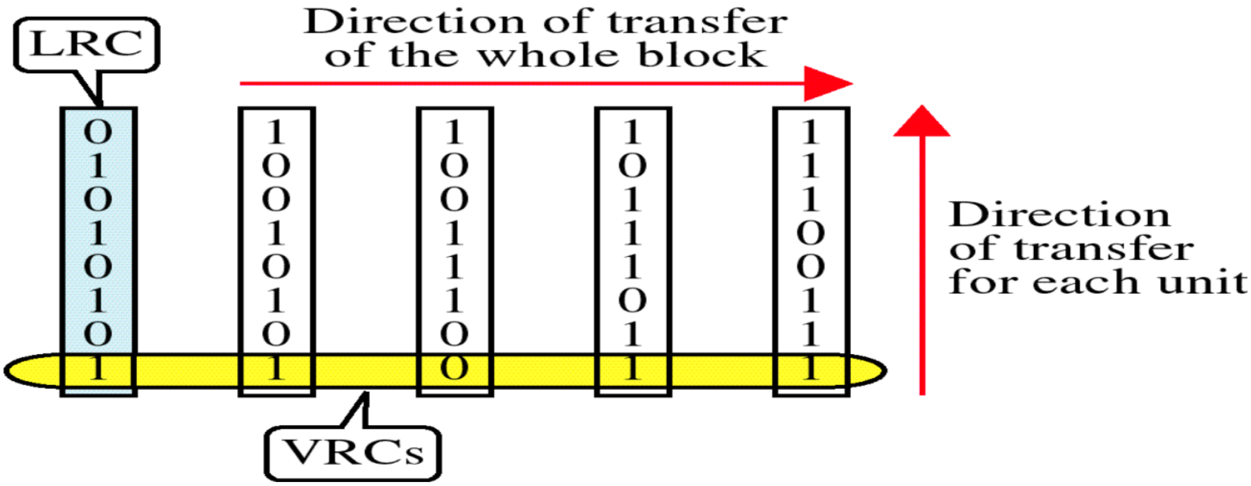
<----- 10101001 00111001 11011101 1110011110101010
(LRC)

However, it is hit by a burst noise of length eight and some bits are corrupted.

<----- 1010*001*1 *1000*1001 11011101 1110011110101010
(LRC)

When the receiver checks the LRC,some of the bits do not follow the even-parity rule and the whole block is discarded(the nonmatching bits are shown in bold.

<----- 1010*001*1 *1000*1001 11011101 1110011110101010
(LRC)



CRC and LRC

Fig 29 CRC and LRC

Cyclic redundancy check(CRC)

CRC ,the most powerful of the redundancy checking techniques, is based on binary division. Most powerful of the redundancy checking techniques is the cyclic redundancy check (CRC). This method is based on the binary division. In CRC, the desired sequence of redundant bits are generated and is appended to the end of data unit. It is also called as CRC reminder. So that the resulting data unit becomes exactly divisible by a predetermined binary number.At its destination, the incoming data unit is divided by the same number. If at this step there is no remainder then the data unit indicates that the data unit has been damaged in transit and therefore must be rejected.

The redundancy bits used by CRC are derived by dividing the data unit by a predetermined divisor; the remainder is the CRC.To be valid, a CRC must have two qualities: It must have exactly one less bit than the divisor, and appending it to the end of the data string must make the resulting bit sequence exactly divisible by the divisor.

The following figure shows the process:

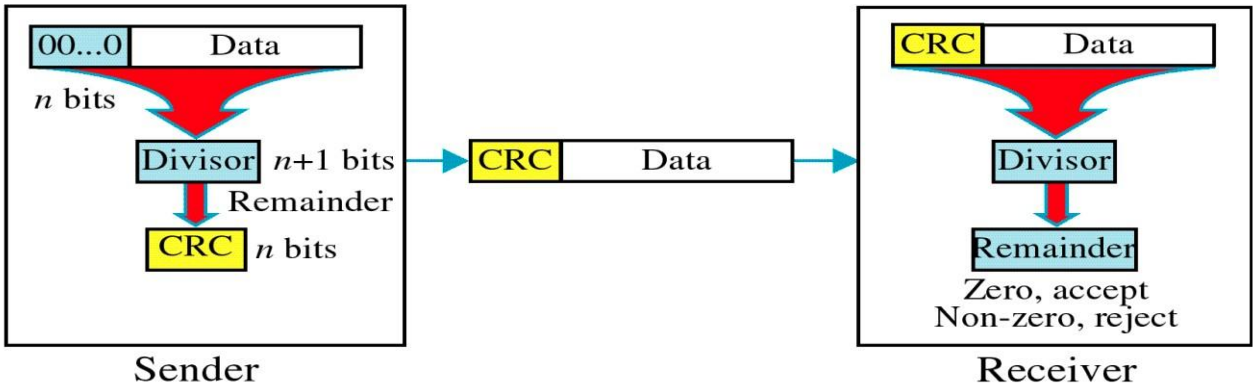


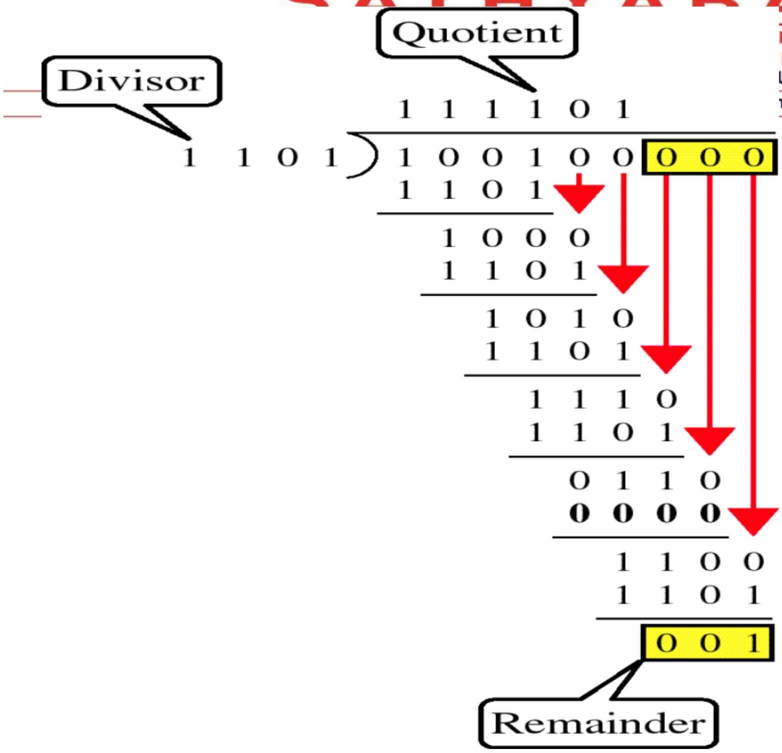
Fig 30 CRC generator and checker

Step1:Astringof0‘sisappendedtothedataunit.Itisnbitslong.Thenumbernis1lessif- number of bits in the predetermined divisor which is n + 1bits.

Step 2: The newly generated data unit is divided by the divisor, using a process called as binary division. The remainder resulting from this division is the CRC.

Step 3: the CRC of n bits derived in step 2 replaces the appended 0‘s at the data unit. Note that the CRC may consist of all 0‘s.

The data unit arrives at the receiver data first, followed by the CRC. The receiver treats the whole string as a unit and divides it by the same divisor that was used the CRC remainder. If the string arrives without error, the CRC checker yields a remainder of zero, the data unit passes. If the string has been changed in transit, the division yields zero remainder and the data unit does not pass.



```
            1 1 1 1 0 1
         ________________
1 1 0 1 ) 1 0 0 1 0 0 0 0 0
          1 1 0 1
          -------
            1 0 0 0
            1 1 0 1
            -------
              1 0 1 0
              1 1 0 1
              -------
                1 1 1 0
                1 1 0 1
                -------
                  0 1 1 0
                  0 0 0 0
                  -------
                    1 1 0 0
                    1 1 0 1
                    -------
                      0 0 1
```

Fig 31 CRC checker Function

A CRC checker functions does exactly as the generator does. After receiving the data appended with the CRC, it does the samemodulo-2 division. If the remainder is all 0‘s, the CRC is dropped and the data is accepted: otherwise, the received stream of bits is discarded and data is resent.

Performance:

CRC is a very effective error detection method. If the divisor is chosen according to the previously mentioned rules,

1. CRC can detect all burst errors that affect an odd number of bits.

2. CRC can detect all burst errors of length less than or equal to the degree of the polynomial

3. CRC can detect, with a very high probability, burst errors of length greater than the degree of the polynomial.

Checksum

A checksum is fixed length data that is the result of performing certain operations on thedata to be sent from sender to the receiver. The sender runs the appropriate checksum algorithm to compute the checksum of the data, appends it as a field in the packet that contains the data to be sent, as well as various headers.

When the receiver receives the data, the receiver runs the same checksum algorithm to compute a fresh checksum. The receiver compares this freshly computed checksum with the checksum that was computed by the sender. If the two checksum matches, the receiver of the data is assured that the data has not changed during the transit. checksum is used by the higher-layer protocols(TCP/IP) for error detection.

To calculate a checksum:

a. Divide the data into sections.
b. Add the sections together using one's complement arithmetic.
c. Take the complement of the final sum; this is the checksum

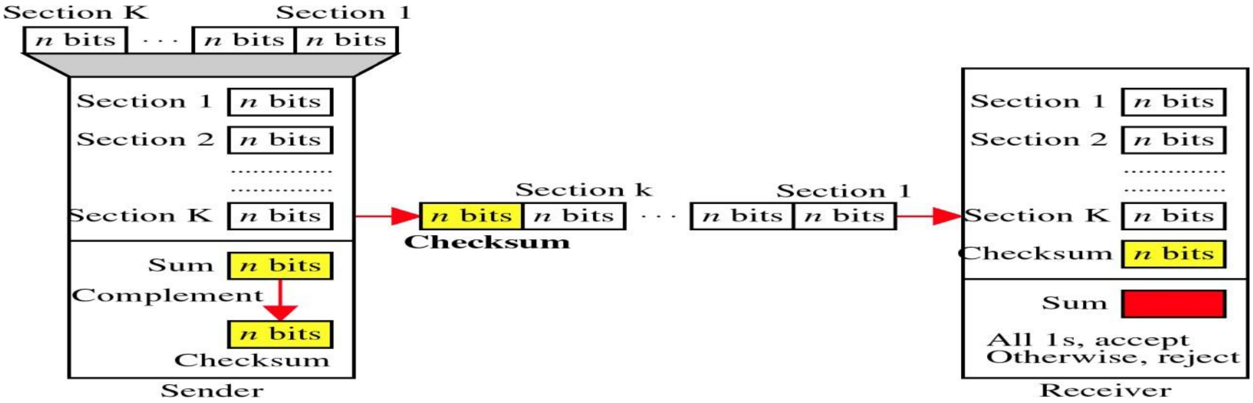


Fig 32 Check Sum

9. Error Correcting Codes

The codes which are used for both error detecting and error correction are called as "Error Correction Codes". The error correction techniques are of two types. They are,

- Single bit error correction
- Burst error correction

The process or method of correcting single bit errors is called "single bit error correction". The method of detecting and correcting burst errors in the data sequence is called "Burst error correction".

Hamming code or Hamming Distance Code is the best error correcting code we use in most of the communication network and digital systems.

Hamming Code

This error detecting and correcting code technique is developed by R.W.Hamming . This code not only identifies the error bit, in the whole data sequence and it also corrects it. This code uses a number of parity bits located at certain positions in the codeword. The number of parity bits depends upon the number of information bits. The hamming code uses the relation between redundancy bits and the data bits and this code can be applied to any number of data bits.

Redundancy Bit

Redundancy means "The difference between number of bits of the actual data sequence and the transmitted bits". These redundancy bits are used in communication system to detect and correct the errors, if any.

Hamming code

In Hamming code, the redundancy bits are placed at certain calculated positions in order to eliminate errors. The distance between the two redundancy bits is called "Hamming distance".

To understand the working and the data error correction and detection mechanism of the hamming code, let's see to the following stages.

Number of parity bits

As we learned earlier, the number of parity bits to be added to a data string depends upon the number of information bits of the data string which is to be transmitted. Number of parity bits will be calculated by using the data bits. This relation is given below.

$2^P >= n + P +1$

Here, n represents the number of bits in the data string.

P represents number of parity bits.

For example, if we have 4 bit data string, i.e. n = 4, then the number of parity bits to be added can be found by using trial and error method. Let's take P = 2, then

$2^P = 2^2 = 4$ and $n + P + 1 = 4 + 2 + 1 = 7$

This violates the actual expression.

So let's try P = 3, then

$2^P = 2^3 = 8$ and $n + P + 1 = 4 + 3 + 1 = 8$

So we can say that 3 parity bits are required to transfer the 4 bit data with single bit error correction.After calculating the number of parity bits required, we should know the appropriate positions to place them in the information string, to provide single bit error correction.In the above considered example, we have 4 data bits and 3 parity bits. So the total codeword to be transmitted is of 7 bits (4 + 3). We generally represent the data sequence from right to left, as shown below.

bit 7, bit 6, bit 5, bit 4, bit 3, bit 2, bit 1, bit 0

The parity bits have to be located at the positions of powers of 2. I.e. at 1, 2, 4, 8 and 16 etc. Therefore the codeword after including the parity bits will be like this

D7, D6, D5, P4, D3, P2, P1

Here P1, P2 and P3 are parity bits. D1 —- D7 are data bits.

Constructing a Bit Location Table

In Hamming code, each parity bit checks and helps in finding the errors in the whole code word. So we must find the value of the parity bits to assign them a bit value. By calculating and inserting the parity bits in to the data bits, we can achieve error correction through Hamming code.

Let's understand this clearly, by looking into an example.

Ex: Encode the data 1101 in even parity, by using Hamming code.

Step 1

Calculate the required number of parity bits.

Let P = 2, then

$2^P = 2^2 = 4$ and $n + P + 1 = 4 + 2 + 1 = 7$.

2 parity bits are not sufficient for 4 bit data.

So let's try P = 3, then

$2^P = 2^3 = 8$ and $n + P + 1 = 4 + 3 + 1 = 8$

Therefore 3 parity bits are sufficient for 4 bit data.

The total bits in the code word are 4 + 3 = 7

Step 2

Constructing bit location table

Step 3

Determine the parity bits.

For P1 : 3, 5 and 7 bits are having three 1's so for even parity, P1 = 1.

For P2 : 3, 6 and 7 bits are having two 1's so for even parity, P2 = 0.
For P3 : 5, 6 and 7 bits are having two 1's so for even parity, P3 = 0.

By entering / inserting the parity bits at their respective positions, codeword can be formed and is transmitted. It is 1100101.

NOTE: If the codeword has all zeros (ex: 0000000), then there is no error in Hamming code.

To represent the binary data in alphabets and numbers, we use alphanumeric codes.

Alpha Numeric Codes

Alphanumeric codes are basically binary codes which are used to represent the alphanumeric data. As these codes represent data by characters, alphanumeric codes are also called "Character codes".

These codes can represent all types of data including alphabets, numbers, punctuation marks and mathematical symbols in the acceptable form by computers. These codes are implemented in I/O devices like key boards, monitors, printers etc.
In earlier days, punch cards are used to represent the alphanumeric codes.

They are

- MORSE code
- BAUDOT code
- HOLLERITH code
- ASCII code
- EBCDI code
- UNICODE

MORSE Code

At the starting stage of computer and digital electronics era, Morse code is very popular and most used code. This was invented by Samuel F.B.Morse, in 1837. It was the first ever telegraphic code used in telecommunication. It is mainly used in Telegraph channels, Radio channels and in air traffic control units.

BOUDOT Code

This code is invented by a French Engineer Emile Baudot, in 1870. It is a 5 unit code, means it uses 5 elements to represent an alphabet. It is also used in Telegraph networks to transfer Roman numeric.

HOLLERITH Code

This code is developed by a company founded by Herman Hollerith in 1896. The 12 bit code used to punch cards according to the transmitting information is called "Hollerith code".

ASCII CODE

ASCII means American Standard Code for Information Interchange. It is the world's most popular and widely used alphanumeric code. This code was developed and first published in 1967. ASCII code is a 7 bit code that means this code uses 27 = 128 characters. This includes

26 lower case letters (a – z), 26 upper case letters (A – Z), 33 special characters and symbols (like ! @ # $ etc), 33 control characters (* – + / and % etc) and 10 digits (0 – 9).

In this 7 bit code we have two parts, the leftmost 3 bits and right side 4 bits. The left most 3 bits are known "ZONE bits" and the right side 4 bits are known as "NUMERIC bits"

Example:

If we want to print the name LONDAN, the ASCII code is?

The ASCII-7 equivalent of L = 100 1100

The ASCII-7 equivalent of O = 100 1111

The ASCII-7 equivalent of N = 100 1110

The ASCII-7 equivalent of D = 100 0100

The ASCII-7 equivalent of A = 100 0001

The ASCII-7 equivalent of N = 100 1110

The output of LONDAN in ASCII code is 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 0.

UNICODE

The draw backs in ASCII code and EBCDI code are that they are not compatible to all languages and they do not have sufficient set of characters to represent all types of data. To overcome these drawback this UNICODE is developed. UNICODE is the new concept of all digital coding techniques. In this we have a different character to represent every number. It is the most advanced and sophisticated language with the ability to represent any type of data. SO this is known as "Universal code". It is a 16 bit code, with which we can represent 216 = 65536 different characters. UNICODE is developed by the combined effort of UNICODE consortium and ISO (International organization for Standardization).

EBCDI CODE

EBCDI stands for Extended Binary Coded Decimal Interchange code. This code is developed by IBM Inc Company. It is an 8 bit code, so we can represent 28 = 256 characters by using EBCDI code. This include all the letters and symbols like 26 lower case letters (a – z), 26 upper case letters (A – Z), 33 special characters and symbols (like ! @ # $ etc), 33 control characters (* – + / and % etc) and 10 digits (0 – 9).In the EBCDI code, the 8 bit code the numbers are represented by 8421 BCD code preceded by 1111.

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

School of Computing

Department of Computer Science and Engineering

UNIT - III

Data Communication and Computer Networks-SBS1302

UNIT III

Multiplexing - types of Multiplexing - Multiplexing Application - Telephone systems project 802 - Ethernet - Token Bus - Token Ring FDD IEEE 802.6 - SMDS - Circuit Switching - Packet switching - Message switching Connection oriented and connectionless services.

1. Multiplexing

Multiplexing is the name given to techniques, which allow more than one message to be transferred via the same communication channel. The channel in this context could be a transmission line, *e.g.* a twisted pair or co-axial cable, a radio system or a fibre optic system *etc*.A channel will offer a specified bandwidth, which is available for a time .

Concept of Multiplexing

Multiplexing is a technique by which different analog and digital streams of transmission can be simultaneously processed over a shared link. Multiplexing divides the high capacity medium into low capacity logical medium which is then shared by different streams.Communication is possible over the air (radio frequency), using a physical media (cable) and light (optical fiber). All mediums are capable of multiplexing.When more than one senders tries to send over single medium, a device called Multiplexer divides the physical channel and allocates one to each. On the other end of communication, a De-multiplexer receives data from a single medium and identifies each and send to different receivers.

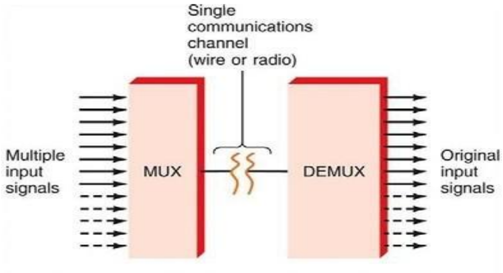


Fig 1 Multiplexing Operation

transmitting two or more signals simultaneously can be accomplished by running multiple cables or setting up one transmitter receiver pair for each channel , but this is an expensive approach. A single cable or radio link can handle multiple signals simultaneously using atechnique known as multiplexing. Multiplexing permits hundreds or even thousands of signals to be combined and transmitted over a single medium.

A device called a multiplexer (often shortened to "mux") combines the input signals into one signal. When the multiplexed signal needs to be separated into its component signal s (for example, when your email is to be delivered to its destination), a device called a demultiplexer (or "demux") isused. Multiplexing was originally developed in the 1800s for telegraphy. Today,multiplexing is widely used in many telecommunications applications, including telephony, Internet communications, digital broadcasting and wireless telephony.

Types of Multiplexing

There are mainly two types of multiplexers, namely analog and digital. They are further divided into Frequency Division Multiplexing (FDM), Wavelength Division Multiplexing (WDM), and Time Division Multiplexing (TDM).

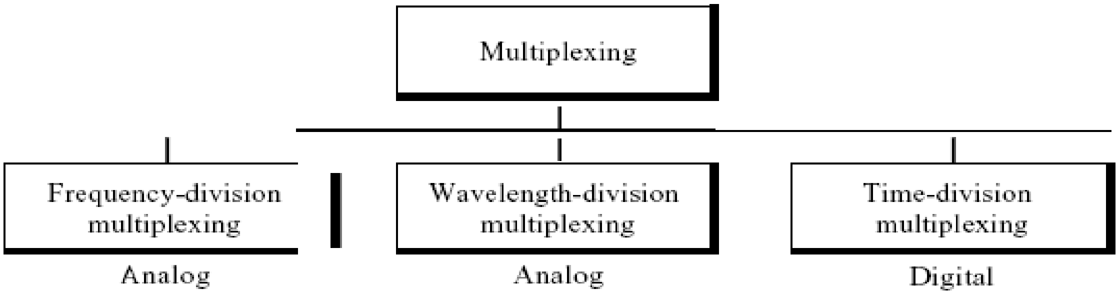

Multiplexing
Frequency-division multiplexing
Wavelength-division multiplexing
Time-division multiplexing
Analog
Analog
Digital

Fig 2 Categories of multiplexing

1) Frequency Division Multiplexing FDM

FDM is derived from AM techniques in which the signals occupy the same physical 'line' but in different frequency bands. Each signal occupies its own specific band of frequencies all the time, *i.e.* the messages share the channel **bandwidth**.

2) Time Division MultiplexingTDM

TDM is derived from sampling techniques in which messages occupy all the channel bandwidth but for short time intervals of time, *i.e.* the messages share the channel **time**.TDM – messages occupy **wide** bandwidth – for short intervals of time. Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

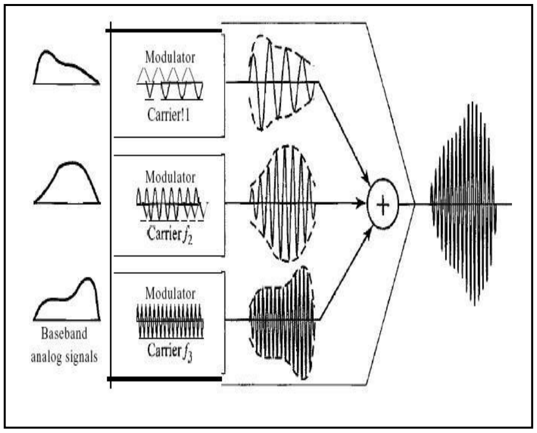


Fig 3 FDM Process

Frequency-Division Multiplexing

Frequency-division multiplexing (FDM) is an analog technique that can be applied when the bandwidth of a link (in hertz) is greater than the combined

bandwidths of the signals to be transmitted. In FOM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth-guard bands-to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original

data frequencies. Fig 4 gives a conceptual view of FDM. In this illustration, the transmission path is divided into three parts, each representing a channel that carries one transmission.

We consider FDM to be an analog multiplexing technique; however, this does not mean that FDM cannot be used to combine sources sending digital signals. A digital signal can be converted to an analog signal before FDM is used to multiplex them.

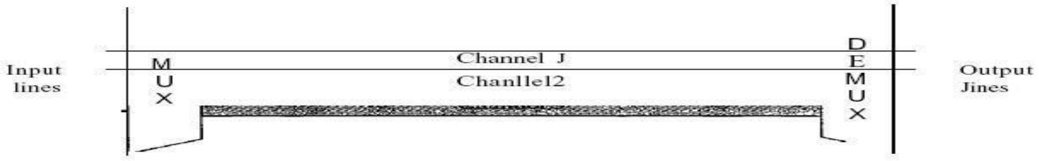


Fig4 FDM

Multiplexing Process

Each source generates a signal of a similar frequency range. Inside the multiplexer, these similar signals modulates different carrier frequencies *(/1,12,* and *h).* The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.

Demultiplexing Process

The demultiplexer uses a series of filters to decompose the multiplexed signal into its constituent component signals. The individual signals are then passed to a demodulator that separates them from their carriers and passes them to the output lines. Fig 5 is a conceptual illustration of demultiplexing process.

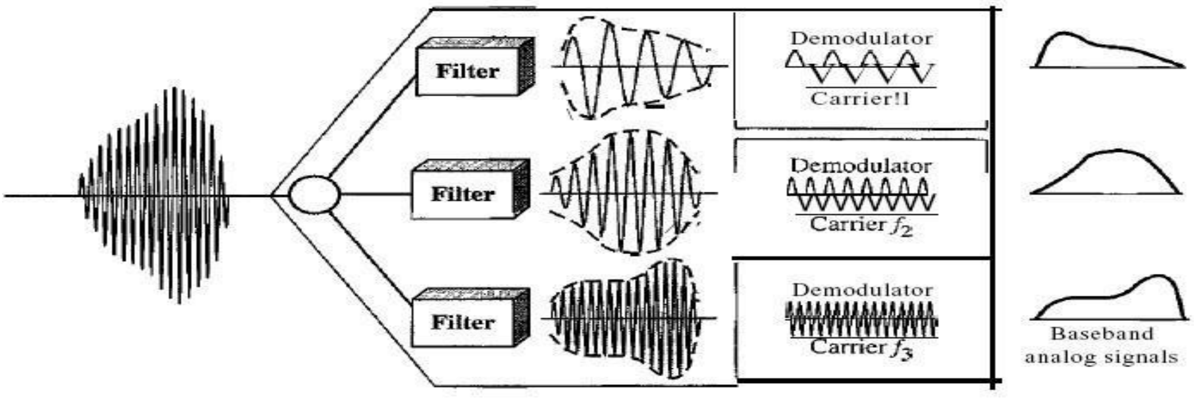


Fig 5 FDM de-multiplexing example

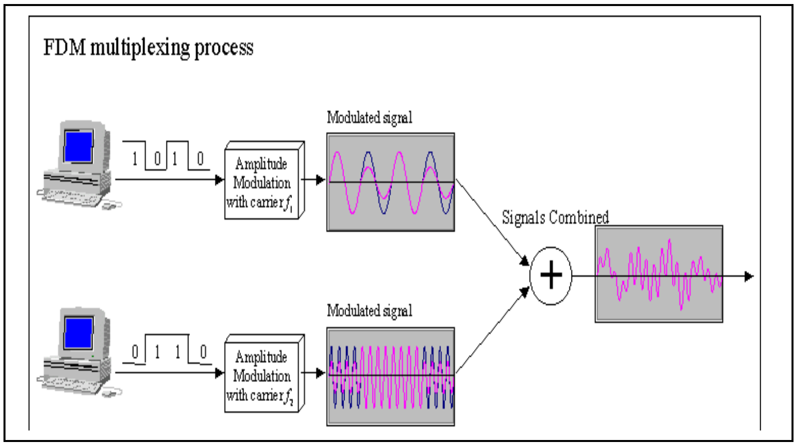


Fig 6 Multiplexing Example

Time Division Multiplexing (TDM): This is possible when data transmission rate of the media is much higher than that of the data rate of the source. Multiple signals can be transmitted ifeach signal is allowed to be transmitted for a definite amount of time. These time slots are so small that all transmissions appear to be in parallel.

Synchronous TDM: Time slots are pre assigned and are fixed. Each source is given it's time slot at every turn due to it. This turn may be once per cycle, or several turns per cycle ,if it has a high data transfer rate, or may be once in a no. of cycles if it is slow. This slot is given even if the source is not ready with data. So this slot is transmitted empty.

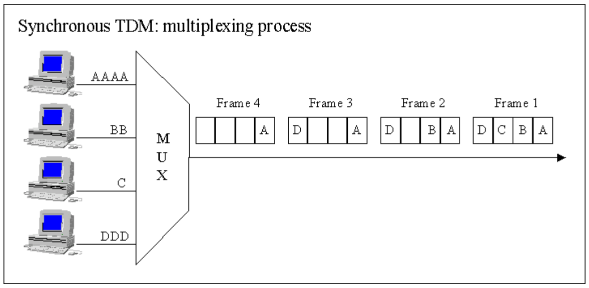


Fig 7

Fig 7 Synchronous TDM

Asynchronous TDM: In this method, slots are not fixed. They are allotted dynamically depending on speed of sources, and whether they are ready for transmission.

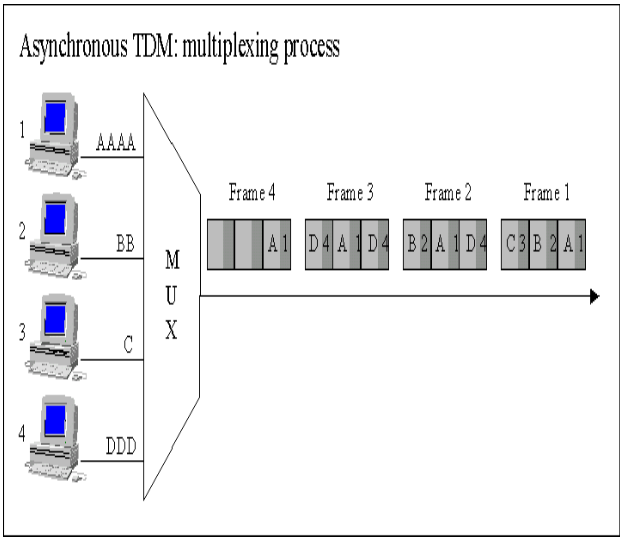


Fig 8Asynchronous TDM

Wavelength-Division Multiplexing

Wavelength-division multiplexing (WDM) is designed to use the high-data-rate capability of fiber-optic cable. The optical fiber data rate is higher than the data rate of metallic transmission cable. Using a fiber-optic cable for one single line wastes the available bandwidth. Multiplexing allows us to combine several lines into one.

WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve optical signals transmitted through fiber-optic channels. The idea is the same: We are combining different signals of different frequencies. The difference is that the frequencies are very high.

Fig 9 gives a conceptual view of a WDM multiplexer and demultiplexer. Very narrow bands of light from different sources are combined to make a wider band of light. At the receiver, the signals are separated by the demultiplexer.

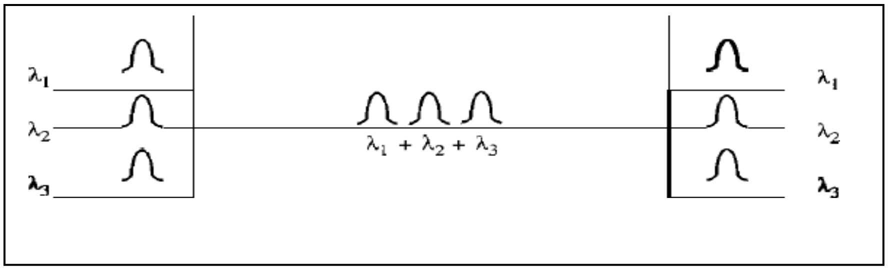


Fig 9Wavelength-division Multiplexing

Although WDM technology is very complex, the basic idea is very simple. We want to combine multiple light sources into one single light at the multiplexer and do the reverse at the demultiplexer. The combining and splitting of light sources are easily handled by a prism. Recall from basic physics that a prism bends a beam of light based on the angle of incidence and the frequency. Using this technique, a multiplexer can be made to combine several input beams of light, each containing a narrow band of frequencies, into one output beam of a wider band of frequencies. A demultiplexer can also be made to reverse the process.

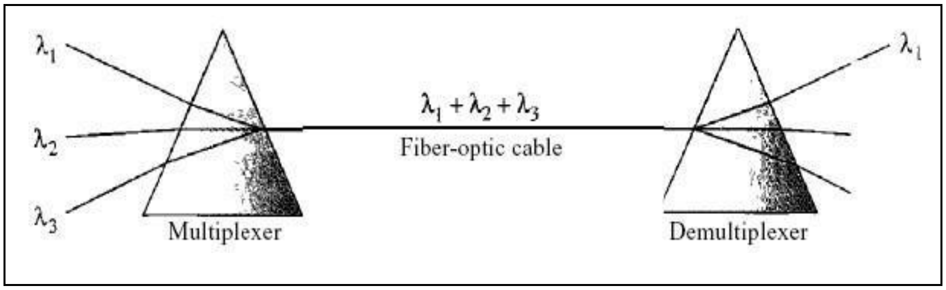


Fig 10 Prisms in wavelength-division multiplexing and demultiplexing

Synchronous Time-Division Multiplexing

Time-division multiplexing (TDM) is a digital process that allows several connections to share the high bandwidth of a linle Instead of sharing a portion of the bandwidth as in FDM, time is shared. Each connection occupies a portion of time in the link. Note that the same link is used as in FDM; here, however, the link is shown sectioned by time rather than by frequency. In the figure, portions of signals 1,2,3, and 4 occupy the link sequentially.

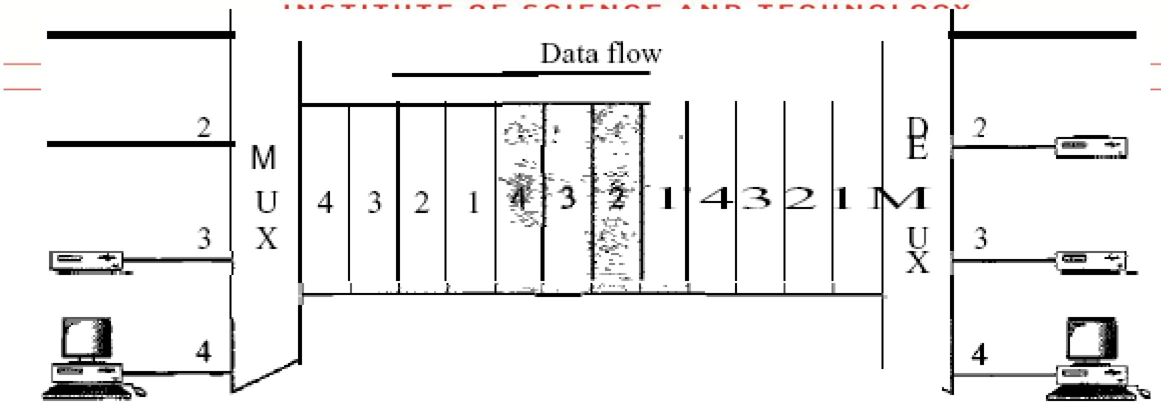


Fig 11 TDM

All the data in a message from source 1 always go to one specific destination, be it 1, 2, 3, or 4. The delivery is fixed and unvarying, unlike switching. We also need to remember thatTDM is, in principle, a digital multiplexing technique. Digital data from different sources are combined into one timeshared link. However, this does not mean that the sources cannot produce analog data; analog data can be sampled, changed to digital data, and then multiplexed by using TDM.

We can divide TDM into two different schemes: synchronous and statistical.

In synchronous TDM, each input connection has an allotment in the output even if it is not sending data.

Time Slots and Frames

In synchronous TDM, the data flow of each input connection is divided into units, where each input occupies one input time slot. A unit can be 1 bit, one character, or one block of data. Each input unit becomes one output unit and occupies one output time slot. However, the duration of an output time slot is *n* times shorter than the duration of an input time slot. If an input time slot is *T* s,the output time slot is*Tin*s,where *n* is the number of connections.Inotherwords,aunitin the ut connection has a shorter duration; it travels faster. Figure shows an example of synchronous TDM where *n* is 3.

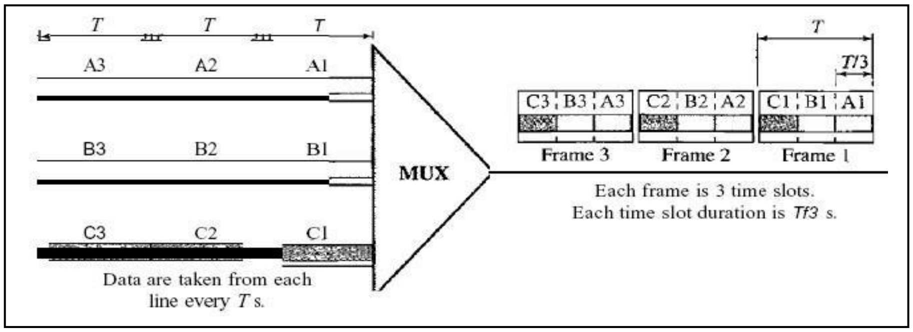


Fig 12 ***Synchronous time-division multiplexing***

In synchronous TDM, a round of data units from each input connection is collected into a frame (we will see the reason for this shortly). If we have *n* connections, a frame is divided into *n* time slots and one slot is allocated for each unit, one for each input line. If the duration of the input unit is *T*, the duration of each slot is *Tin* and the duration of each frame is *T* (unless a frame carries some other information, as we will see shortly).

The data rate of the output link must be *n* times the data rate of a connection to guarantee the flow of data. In Figure 7, the data rate of the link is 3 times the data rate of a connection; likewise, the duration of a unit on a connection is 3 times that of the time slot (duration of a unit on the link). In the figure we represent the data prior to multiplexing as 3 times the size of the data after multiplexing. This is just to convey the idea that each unit is 3 times longer in duration before multiplexing than after. Time slots are grouped into frames. A frame consists of one complete cycle of time slots, with one slot dedicated to each sending device. In a system with *n* input lines, each frame has *n* slots, with each slot allocated to carrying data from a specific input line.

Interleaving

TDM can be visualized as two fast-rotating switches, one on the multiplexing side and the other on the demultiplexing side. The switches are synchronized and rotate at the same speed, but in opposite directions. On the multiplexing side, as the switch opens in front of a connection,thatconnection has the opportunity to send a unit onto the path. This process is called **interleaving.** On the demultiplexing side, as the switch opens in front of a connection, that connection has the opportunity to receive a unit from thepath.

In this figure, we assume that no switching is involved and that the data from the first connection at the multiplexer site go to the first connection at the demultiplexer

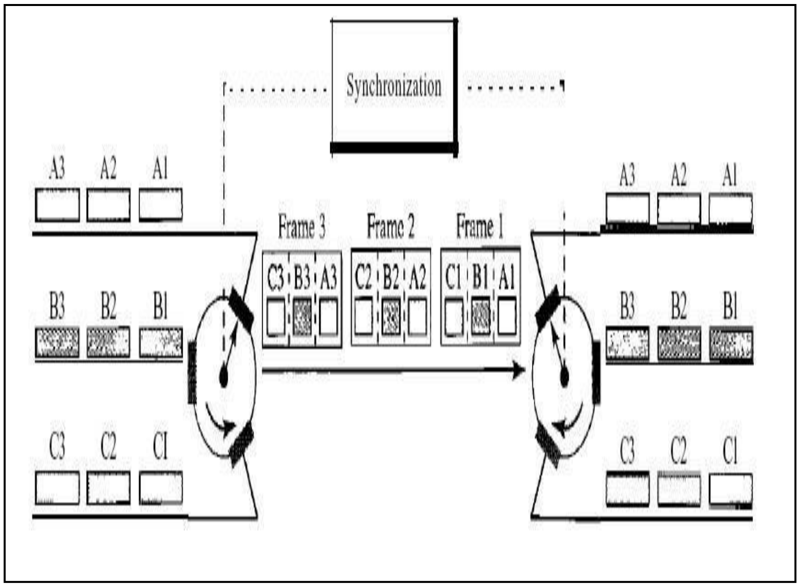


Fig 13 Interleaving

Statistical Time-Division Multiplexing

Addressing

Figure a also shows a major difference between slots in synchronous TDM and statistical TDM. An output slot in synchronous TDM is totally occupied by data; in statistical TDM, a slot needs to carry data as well as the address of the destination.

In synchronous TDM, there is no need for addressing; synchronization and preassigned relationships between the inputs and outputs serve as an address. We know, for example, that input 1 always goes to input 2. If the multiplexer and the demultiplexer are synchronized, this is guaranteed. In statistical multiplexing, there is no fixed relationship between the inputs and outputs because there are no pre assigned or reserved slots. We need to include the address of the receiver inside each slot to show where it is to be delivered. The addressing in its simplest form can be *n* bits to define *N* different output lines with *n* =10g2 *N*. For example, for eight different output lines, we need a 3-bitaddress.

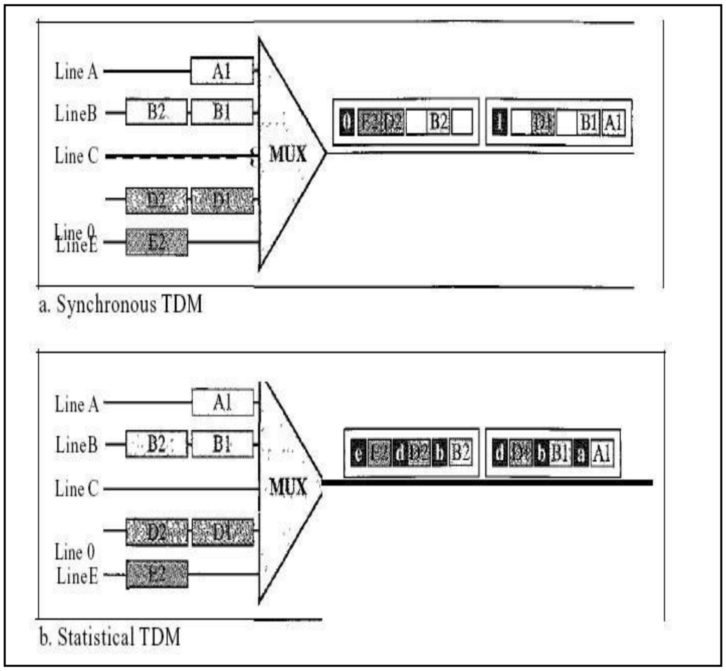


Fig 14 TDM slot comparison

Slot Size

Since a slot carries both data and an address in statistical TDM, the ratio of the data size to address size must be reasonable to make transmission efficient. For example, it would be inefficient to send 1 bit per slot as data when the address is 3 bits. This would mean an overhead of 300 percent. In statistical TDM, a block of data is usually many bytes while the address is just a few bytes.

No Synchronization Bit

There is another difference between synchronous and statistical TDM, but this time it is at the frame level. The frames in statistical TDM need not be synchronized, so we do not need synchronization bits.

Bandwidth

In statistical TDM, the capacity of the link is normally less than the sum of the capacities of each channel. The designers of statistical TDM define the capacity of the link based on the statistics of the load for each channel. If on average only *x* percent of the input slots are filled, the capacity of the link reflects this. Of course, during peak times, some slots need to wait.

2. Applications of Multiplexers

A Multiplexer is used in numerous applications like, where multiple data can be transmitted using a single line.

Communication System – A Multiplexer is used in communication systems, which has a transmission system and also a communication network. A Multiplexer is used to increase the efficiency of the communication system by allowing the transmission of data such as audio & video data from different channels via cables and single lines.

Computer Memory – A Multiplexer is used in computer memory to keep up a vast amount of memory in the computers, and also to decrease the number of copper lines necessary to connect the memory to other parts of the computer.

Telephone Network – A multiplexer is used in telephone networks to integrate the multiple audio signals on a single line of transmission. In a telephone network, the multiple audio signals are brought into a single line and transmitted with the implementation of a Mux. By this way, the numerous audio signals are made isolated and ultimately the recipient will receive the required audio signals.

Computer System of a Satellite Transmission- Mux is used for the data signals to be transmitted from spacecraft or computer system of a satellite to the earth by means of GPS.

3. IEEE Standards

The IEEE has subdivided the data link layer into two sublayers: logical link control (LLC) and media access control (MAC). IEEE has also created several physical layer standards for different LAN protocols.

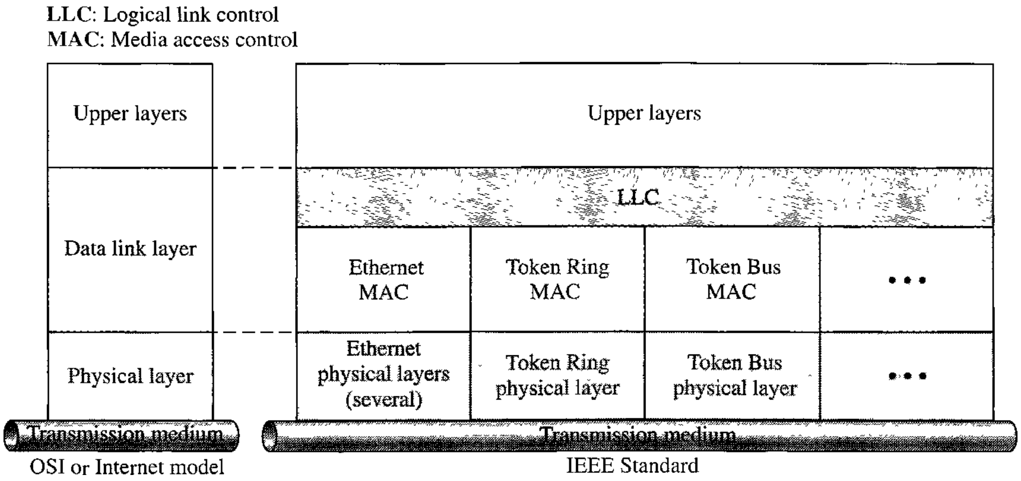


Fig 15 IEEE Standards

Data Link Layer

The data link layer in the IEEE standard is divided into two sublayers: LLC and MAC.

Logical Link Control (LLC)

In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control. Framing is handled in both the LLC sublayer and the MAC sublayer.

The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer, which provides different protocols for different LANs. A single LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent.

Framing LLC defines a protocol data unit (PDU) that is somewhat similar to that of HDLC. The header contains a control field like the one in HDLC; this field is used for flow and error control. The two other header fields define the upper-layer protocol at the source and destination that uses LLC. These fields are called the destination service access point (DSAP) and the source service access point (SSAP). The other fields defined in a typical data link control protocol such as HDLC are moved to the MAC sublayer. In other words, a frame defined in HDLC is divided into a PDU at the LLC sublayer and a frame at the MAC sublayer, as shown in figure.

Need for LLC The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services. For example, if a LAN or several LANs are used in an isolated system, LLC may be needed to provide flow and error control for the application layer protocols. However, most upper-layer protocols such as IP, do not use the services of LLC.

Media Access Control (MAC)

IEEE Project 802 has created a sub layer called media access control that defines the specific access method for each LAN. For example, it defines CSMA/CD as the media access method for Ethernet LANs and the token-passing method for Token Ring and Token Bus LANs. Part of the framing function is also handled by the MAC layer.In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.

Physical Layer

The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specification for each Ethernet implementations.

Key features of LANs are summarized below:

- Limited geographical area – which is usually less than 10 Km and more than 1m.
- High Speed – 10 Mbps to 1000 Mbps (1 Gbps) and more
- High Reliability – 1 bit error in 10^{11}bits.
- Transmission Media – Guided and unguided media, mainly guided media is used; except in a situation where infrared is used to make a wireless LAN in a room.
- Topology – It refers to the ways in which the nodes are connected. There are various topologies used.

Medium-Access Control Techniques –Some access control mechanism is needed to decide which station will use the shared medium at a particular point in time.

For the fulfillment of the abovementioned goals, the committee came up with a bunch of LAN standards collectively known as IEEE 802 LANs. To satisfy diverse requirements, the standard includes CSMA/CD, Token bus, Token

Ring medium access control techniques along with different topologies. All these standards differ at the physical layer and MAC sublayer, but are compatible at the data link layer.

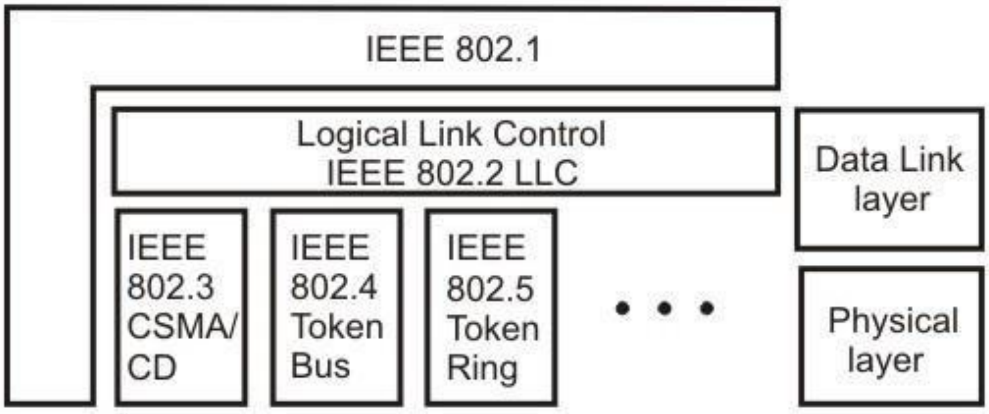


Fig 16 IEEE 802 Legacy LANs

The **802.1** sublayer gives an introduction to set of standards and gives the details of the interface primitives. It provides relationship between the OSI model and the 802 standards. The **802.2** sublayer describes the **LLC** (logical link layer), which is the upper part of the data link layer. LLC facilitate error control and flow control for reliable communication. It appends a header containing sequence number and acknowledgement number. And offers the following three types of services:

- Unreliable datagram service
- Acknowledged datagram service
- Reliable connection oriental service

The standards 802.3, 802.4 and 802.5 describe three LAN standards based on the CSMA/CD, token bus and token ring, respectively. Each standard covers the physical layer and MAC sublayer protocols. In the following sections we shall focus on these three LAN standards.

4. IEEE 802.3 and Ethernet

Ethernet - A Brief History

The original Ethernet was developed as an experimental coaxial cable network in the 1970s by Xerox Corporation to operate with a data rate of 3 Mbps using a carrier sense multiple access collision detection (CSMA/CD) protocol for LANs with sporadic traffic requirements. Success with that project attracted early attention and led to the 1980 joint development of the 10-Mbps Ethernet Version 1.0 specification by the three-company consortium: Digital Equipment Corporation, Intel Corporation, and Xerox Corporation.

The original IEEE 802.3 standard was based on, and was very similar to, the Ethernet Version 1.0 specification. The draft standard was approved by the 802.3 working group in 1983 and was subsequently published as an official standard in 1985 (ANSI/IEEEStd.

802.3-1985). From then onwards, the term *Ethernet* refers to the family of local-area network (LAN) products covered by the IEEE 802.3 standard that defines what is commonly known as the CSMA/CD protocol. Three data rates are currently defined for operation over optical fiber and twisted-pair cables:

- 10 Mbps—10Base-TEthernet
- 100 Mbps—FastEthernet
- 1000 Mbps—GigabitEthernet

Ethernet has survived as the major LAN technology (it is currently used for approximately 85 percent of the world's LAN-connected PCs and workstations) because its protocol has the following characteristics:

- ▪ It is easy to understand, implement, manage, and maintain It allows low-cost network implementations
- ▪ It provides extensive topological flexibility for network installation
- ▪ It guarantees successful interconnection and operation of standards-compliant products, regardless of manufacturer

Ethernet Architecture

Ethernet architecture can be divided into two layers:

➢ **Physical layer:** this layer takes care of following functions.

- Encoding and decoding
- Collision detection
- Carrier sensing
- Transmission and receipt

➢ **Data link layer:** Following are the major functions of this layer.

- Station interface

- Data Encapsulation/De capsulation
- Link management
- Collision Management

STANDARD ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations: Standard Ethernet (10 t Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps), and Ten-Gigabit Ethernet (10 Gbps), as shown in the figure:

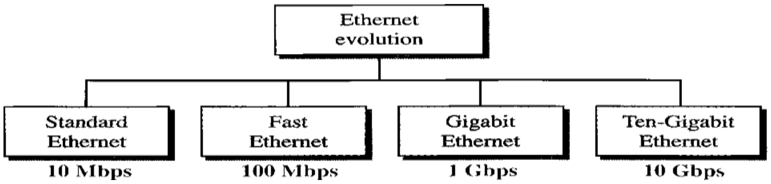


Fig 17 Types of Ethernet

MAC Sublayer

In Standard Ethernet, the MAC sub layer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRC. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers. The format of the MAC frame is shown in the figure.

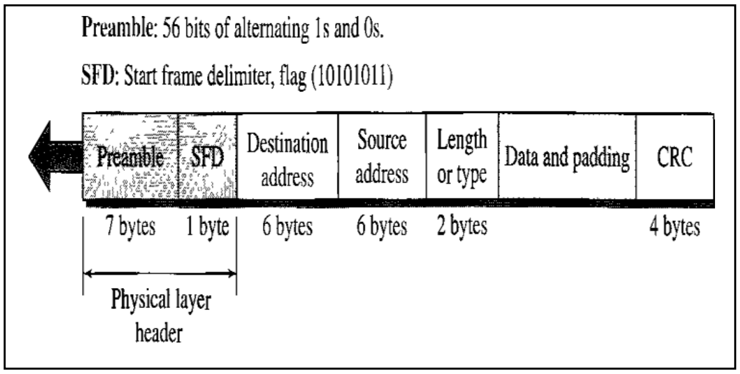


Fig 18 a) MAC Frame

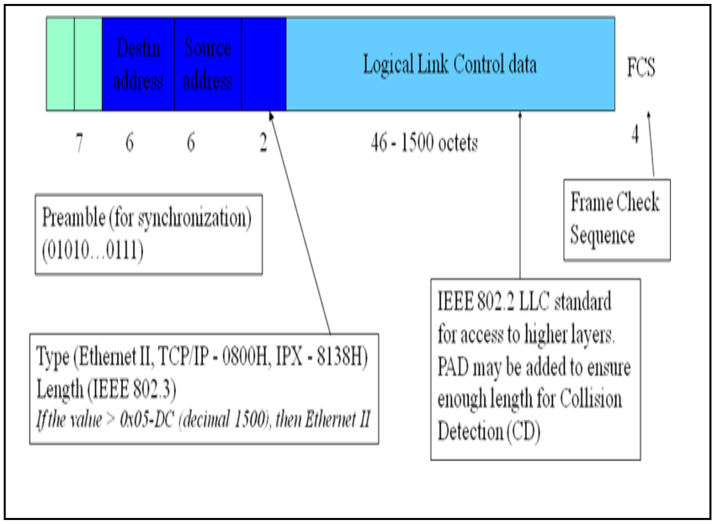

Destn address
Source address
Logical Link Control data
FCS
7
6
6
2
46 - 1500 octets
4
Preamble (for synchronization) (01010…0111)
Frame Check Sequence
IEEE 802.2 LLC standard for access to higher layers. PAD may be added to ensure enough length for Collision Detection (CD)
Type (Ethernet II, TCP/IP - 0800H, IPX - 8138H) Length (IEEE 802.3)
If the value > 0x05-DC (decimal 1500), then Ethernet II

Fig 18 b) MAC Frame Description

- **Preamble**. The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The preamble is actually added at the physical layer and is not (formally) part of the frame.
- **Start frame delimiter (SFD).** The second field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.
- **Destination address (DA)**. The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.
- **Source address (SA)**. The SA field is also 6 bytes and contains the physical address of the sender of the packet.
- **Length or type**. This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field.
- **Data**. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500bytes.

- **CRC**. The last field contains error detection information, in this case aCRC-32.

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in figure.

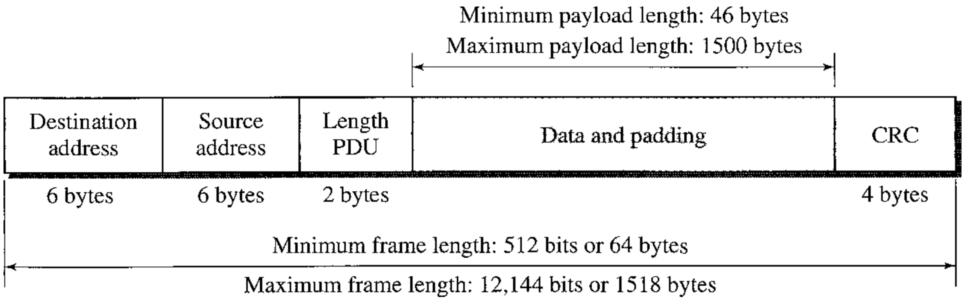

Minimum payload length: 46 bytes
Maximum payload length: 1500 bytes
Destination address
Source address
Length PDU
Data and padding
CRC
6 bytes
6 bytes
2 bytes
4 bytes
Minimum frame length: 512 bits or 64 bytes
Maximum frame length: 12,144 bits or 1518 bytes

Fig 19 Frame Length

The minimum length restriction is required for the correct operation of CSMA/CD. An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 - 18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference.

The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes. The maximum length restriction has two historical reasons. First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer. Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Addressing

Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a 6-byte physical address. As shown in the figure, the Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.

Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address--the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. The following figure shows how to distinguish a unicast address from a multicast address. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.

A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one-to-one. A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one4o-many. The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty-eight ls.

Access Method: CSMA/CD

Standard Ethernet uses 1-persistent CSMA/CD

Slot Time In an Ethernet network, the round-trip time required for a frame to travel from one end of a maximum-length network to the other plus the time needed to send the jam sequence is called the slot time.

Slot time = round-trip time + time required to send the jam sequence

The slot time in Ethernet is defined in bits. It is the time required for a station to send 512 bits. This means that the actual slot time depends on the data rate; for traditional 10- Mbps Ethernet it is51.2µs.

Slot Time and Collision The choice of a 512-bit slot time was not accidental. It Waschosen to allow the proper functioning of CSMA/CD. To understand the situation, let us consider two cases.

In the first case, we assume that the sender sends a minimum-size packet of 512 bits. Before the sender can send the entire packet out, the signal travels through the network and reaches the end of the network. If there is another signal at the end of the network (worst case), a collision occurs. The sender has the opportunity to abort the sending of the frame and to send a jam sequence to inform other stations of the collision. The roundtrip time plus the time required to send the jam sequence should be less than the time needed for the sender to send the minimum frame, 512 bits. The sender needs to be aware of the collision before it is too late, that is, before it has sent the entire frame.

In the second case, the sender sends a frame larger than the minimum size (between 512 and 1518 bits). In this case, if the station has sent out the first 512 bits and has not hearda collision, it is guaranteed that collision will never occur during the transmission of this frame. The reason is that the signal will reach the end of the network in less than one- half the slot time. If all stations follow the CSMA/CD protocol, they have already sensed the existence of the signal (carrier) on the line and have refrained from sending. If they sent a signal on the line before one- half of the slot time expired, a collision has occurred and the sender has sensed the collision. In other words, collision can only occur during the first half of the slot time, and if it does, it can be sensed by the sender during the slot time. This means that after the sender sends the first 512 bits, it is guaranteed that collision will not occur during the transmission of this frame. The medium belongs to the sender, and no other station will use it. In other words, the sender needs to listen for a collision only during the time the first 512 bits aresent.

Slot Time and Maximum Network Length There is a relationship between the slot time and the maximum length of the network (collision domain). It is dependent on the propagation speed of the signal in the particular medium. In most transmission media, the signal propagates at 2 x 108 m/s (two-thirds of the rate for propagation in air). For traditional Ethernet, wecalculate

$$\text{MaxLength} = \text{PropagationSpeed} \times \frac{\text{SlotTime}}{2}$$

$$\text{MaxLength} = (2 \times 10^{8}) \times (51.2 \times 10^{-6} / 2) = 5120 \text{ m}$$

Of course, we need to consider the delay times in repeaters and interfaces, and the time required to send the jam sequence. These reduce the maximum-length of a traditional Ethernet network to 2500 m, just 48 percent of the theoretical calculation.

MaxLength = 2500 m

Physical Layer

The Standard Ethernet defines several physical layer implementations; four of the most common, are shown in figure.

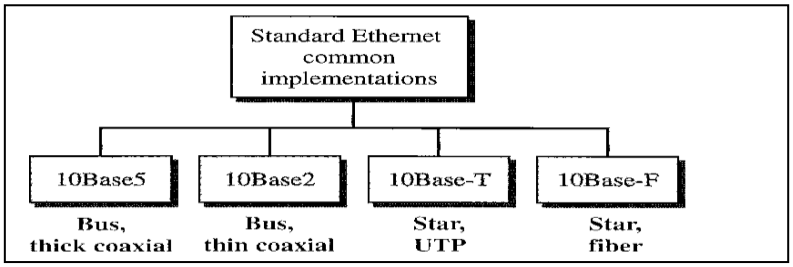


Fig 20 Ethernet Description

Because Ethernet devices implement only the bottom two layers of the OSI protocol stack, they are typically implemented as network interface cards (NICs) that plug into the host device's motherboard, or presently built-in in the motherboard. The naming convention is a concatenation of three terms indicating the transmission rate, the transmission method, and the media type/signal encoding. Consider for example, 10Base-T. where 10 implies transmission rate of 10 Mbps, Base represents that it uses baseband signaling, andTrefers to twisted-pair cables as transmission media. Various standards are discussed below:

Encoding and Decoding

All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the received signal is interpreted as Manchester and decoded into data. The figure shows the encoding scheme for StandardEthernet.

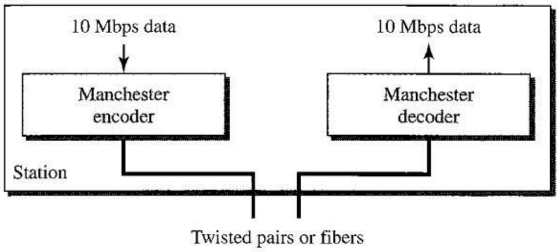


Fig 21 Encoding Scheme

10Base5: Thick Ethernet

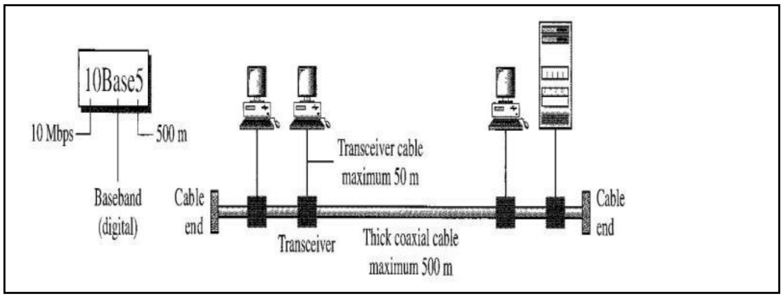
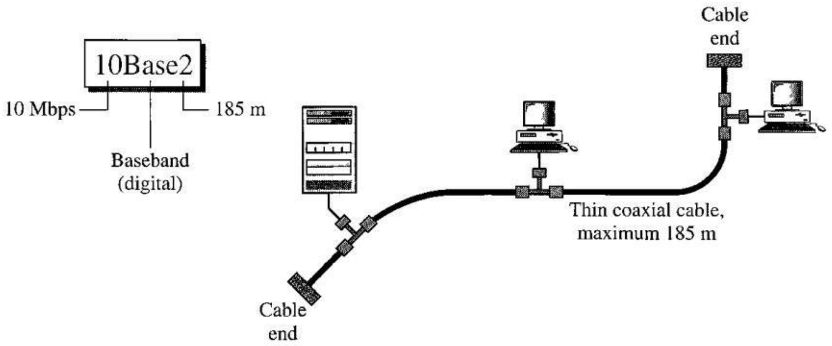


Fig 22 Thick Ethernet

10Base5 was the first Ethernet specification to use a bus topology with an external transceiver (transmitter/receiver) connected via a tap to a thick coaxial cable. The transceiver is responsible for transmitting, receiving, and detecting collisions.

The transceiver is connected to the station via a transceiver cable that provides separate paths for sending and receiving. This means that collision can only happen in the coaxial cable.

The maximum length of the coaxial cable must not exceed 500 m, otherwise, there is excessive degradation of the signal. If a length of more than 500 m is needed, up to five segments, each a maximum of 500-meter, can be connected using repeaters.



10Base2: Thin Ethernet

Fig 23 Thin Ethernet

10Base2 also uses a bus topology, but the cable is much thinner and more flexible. The cable can be bent to pass very close to the stations. In this case, the transceiver is normally part of the network interface card (NIC), which is installed inside the station.

Note that the collision here occurs in the thin coaxial cable. This implementation is more cost effective than 10Base5 because thin coaxial cable is less expensive than thick coaxial and the tee connections are much cheaper than taps. Installation is simpler because the thin coaxial cable is very flexible. However, the length of each segment cannot exceed 185 m (close to 200 m) due to the high level of attenuation in thin coaxial cable.

10Base- T: Twisted-Pair Ethernet

The third implementation is called 10Base-T or twisted-pair Ethernet. 10Base-T uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable.

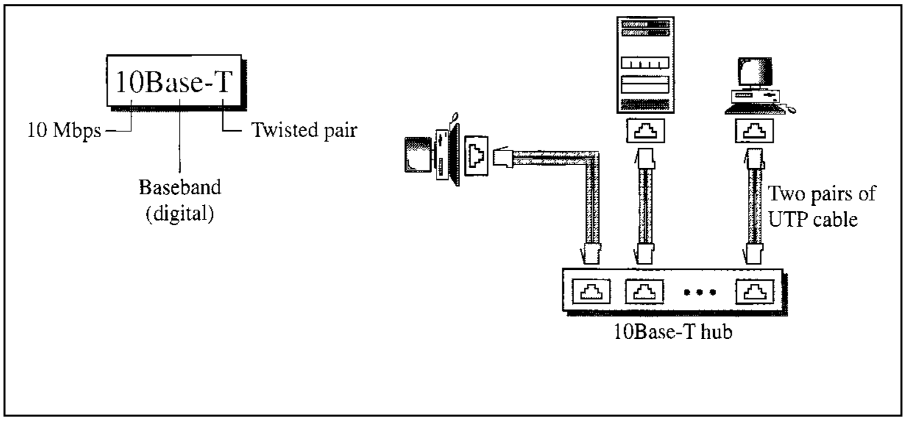


Fig 24 Twisted Pair

Note that two pairs of twisted cable create two paths (one for sending and one for receiving) between the station and the hub. Any collision here happens in the hub. Compared to 10Base5 or 10Base2, we can see that the hub actually replaces the coaxial cable as far as a collision is concerned. The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable.

10Base-F: Fiber Ethernet

10Base-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables.

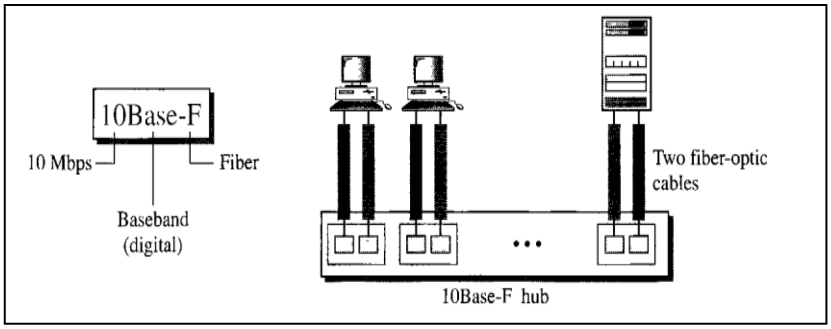


Fig 25 Fiber Ethernet

No Need for CSMA/CD

In full-duplex switched Ethernet, there is no need for the CSMA/CD method. In a full-duplex switched Ethernet, each station is connected to the switch via two separate links. Each station or switch can send and receive independently without worrying about collision. Each link is a point-to-point dedicated path between the station and the switch. There is no longer a need for carrier sensing; there is no longer a need for collision detection. The job of the MAC layer becomes much easier. The carrier sensing and collision detection functionalities of the MAC sub layer can be turned off.

MAC Control Layer

Standard Ethernet was designed as a connectionless protocol at the MAC sublayer. There is no explicit flow control or error control to inform the sender that the frame has arrived at the destination without error. When the receiver receives the frame, it does not send any positive or negative acknowledgment.

To provide for flow and error control in full-duplex switched Ethernet, a new sublayer, called the MAC control, is added between the LLC sublayer and the MAC sublayer.

FAST ETHERNET

Fast Ethernet was designed to compete with LAN protocols such as FDDI or Fiber Channel (or Fibre Channel, as it is sometimes spelled). IEEE created Fast Ethernet under the name 802.3u. Fast Ethernet is backward-compatible with Standard Ethernet, but it can transmit data 10 times faster at a rate of 100 Mbps. The goals of Fast Ethernet can be summarized as follows:

1. Upgrade the data rate to 100Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bitaddress.
4. Keep the same frame format.
5. Keep the same minimum and maximum frame lengths.

MAC Sub layer

A main consideration in the evolution of Ethernet from 10 to 100 Mbps was to keep the MAC sub layer untouched. However, a decision was made to drop the bus topologies and keep only the star topology. For the star topology, there are two choices, as we saw before: half duplex and full duplex. In the half-duplex approach, the stations are connected via a hub; in the full- duplex approach, the connection is made via a switch with buffers at each port.
The access method is the same (CSMA/CD) for the half-duplex approach; for full- duplex Fast Ethernet, there is no need for CSMA/CD. However, the implementations keep CSMA/CD for backward compatibility with Standard Ethernet.

GIGABIT ETHERNET

The need for an even higher data rate resulted in the design of the Gigabit Ethernet protocol (1000 Mbps). The IEEE committee calls the Standard 802.3z. The goals of the Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 1Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bitaddress.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. To support auto negotiation as defined in Fast Ethernet.

Ten-Gigabit Ethernet

The IEEE committee created Ten-Gigabit Ethernet and called it Standard 802.3ae.

The goals of the Ten-Gigabit Ethernet design can be summarized as follows:

1. Upgrade the data rate to 10Gbps.
2. Make it compatible with Standard, Fast, and Gigabit Ethernet.
3. Use the same 48-bitaddress.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. Allow the interconnection of existing LANs into a metropolitan area network (MAN) or a wide area network(WAN).

7. Make Ethernet compatible with technologies such as Frame Relay and ATM.

MAC Sublayer

Ten-Gigabit Ethernet operates only in full duplex mode which means there is no need for contention; CSMA/CD is not used in Ten-Gigabit Ethernet.

Physical Layer

The physical layer in Ten-Gigabit Ethernet is designed for using fiber-optic cable over long distances. Three implementations are the most common: 10GBase-S, 10GBase-L, and 10GBase- E.

6. Token Ring (IEEE 802.5)

Token Ring: A Brief History

Originally, IBM developed Token Ring network in the 1970s. It is still IBM's primary local-area network (LAN) technology. The related IEEE 802.5 specification is almost identical to and completely compatible with IBM's Token Ring network. In fact, the IEEE 802.5 specification was modeled after IBM Token Ring, and on the same lines. The term *Token Ring* is generally used to refer to both IBM's Token Ring network andIEEE802.5 networks.

Before going into the details of the Token Ring protocol, let's first discuss the motivation behind it. As already discussed, the medium access mechanism used by Ethernet (CSMA/CD) may results in collision. Nodes attempt to a number of times before they can actually transmit, and even when they start transmitting there are chances to encounter collisions and entire transmission need to be repeated. And all this become worse one the traffic is heavy i.e. all nodes have some data to transmit. Apart from this there is no way to predict either the occurrence of collision or delays produced by multiple stations attempting to capture the link at the same time. So all these problems with the Ethernet gives way to an alternate LAN technology, Token Ring.

Token Ring and IEEE802.5 are based on token passing MAC protocol with ring topology. They resolve the uncertainty by giving each station a turn on by one. Each node takes turns sending the data; each station may transmit data during its turn. The technique that coordinates this turn mechanism is called Token passing; as a Token is passed in the network and the station that gets the token can only transmit. As one node transmits at a time, there is no chance of collision. We shall discuss the detailed operation in next section.

Stations are connected by point-to-point links using repeaters. Mainly these links are of shielded twisted-pair cables. The repeaters function in two basic modes: Listen mode, Transmit mode. A disadvantage of this topology is that it is vulnerable to link or station failure. But a few measures can be taken to take care of it.

Differences between Token Ring and IEEE 802.5

Both of these networks are basically compatible, although the specifications differin someways. IEEE 802.5 does not specify a topology, although virtually all IEEE802.5 implementations are based on the star topology. While IBM's Token Ring network explicitly specifies a star, with all end stations attached to a device called a Multi- Station Access Unit (MSAU).IEEE 802.5 does not specify a media type, although IBM Token Ring network suse twisted-pair wire. The most common local area network alternative to Ethernet is a network technology developed by IBM, called tokenring. Where Ethernet relies on the random gaps between transmissions to regulate access to the medium, token ring implements a strict, orderly access method.

A token-ring network arranges nodes in a logical ring, as shown below. The nodes forward frames in one direction around the ring,removing a frame when it has circled the ring once. The ring initializes by creating a **token**, which is a special type of frame that gives a station permission to transmit. The token circles the ring like any frame until it encounters a station that wishes to transmit data. This station then "captures" the token by replacing the token frame with a data- carrying frame, which encircles the network. Once that data frame returns to the transmitting station, that station removes the data frame, creates a new token and forwards that token on to the next node in the ring.

Token-ring nodes do not look for a carrier signal or listen for collisions; the presence of the token frame provides assurance that the station can transmit a data frame without fear of another station interrupting. Because a station transmits only a single data frame before passing the token along, each station on the ring will get a turn to communicate in a deterministic and fair manner. Token- ring networks typically transmit data at either 4 or 16Mbps.There are few differences in routing information field size of the two.

Token Ring Operation

Token-passing networks move a small frame, called a *token*, around the network. Possession of the token grants the right to transmit. If a node receiving the token has no information to send, it passes the token to the next end station. Each station can hold the token for a maximum period of time.If a station possessing the token does have information to transmit, it seizes the token, alters 1 bit of the token (which turns the token into a start-of-frame sequence), appends the information that it wants to transmit, and sends this information to the next station on the ring. While the information frame is circling the ring, no token is on the network (unless the ring supports early token release), which means that other stations wanting to transmit must wait. Therefore, *collisions cannot occur in Token Ring networks*. If *early token release* is supported, a new token can be released immediately after a frame transmission iscomplete.

The information frame circulates around the ring until it reaches the intended destination station, which copies the information for further processing. The information frame makes a round trip and is finally removed when it reaches the sending station. The sending station can check the returning frame to see whether the frame was seen and subsequently copied by the destination station in error-free form. Then the sending station inserts a new free token on the ring, if it has finished transmission of itspackets.

Unlike CSMA/CD networks (such as Ethernet), token-passing networks are *deterministic*, which means that it is possible to calculate the maximum time that will pass before any end station will be capable of transmitting. Token Ring networks are ideal for applications in which delay must be predictable and robust network operation is important.

Priority System

Token Ring networks use a sophisticated priority system that permits certain user- designated, high-priority stations to use the network more frequently. Token Ring frames have two fields that control priority: *the priority field* and the *reservation field.*Only stations with a priority equal to or higher than the priority value contained in a token can seize that token. After the token is seized and changed to an information frame, only stations with a priority value higher than that of the transmitting station can reserve the token for the next pass around the network. When the next token is generated, it includes the higher priority of the reserving station. Stations that raise a token's priority level must reinstate the previous priority after their transmission is complete.

Ring Maintenance

There are two error conditions that could cause the token ring to break down. One is the *lost token* in which case there is no token the ring, the other is the *busy token* that circulates endlessly. To overcome these problems, the IEEE 802 standard specifies that one of the

stations be designated as 'active monitor'. The monitor detects the lost condition using a timer by *time-out* mechanism and recovers by using a new free token. To detect a circulating busy token, the monitor sets a 'monitor bit' to one on any passing busy token. If it detects a busy token with the monitor bit already set, it implies that the sending station has failed to remove its packet and recovers by changing the busy token toafreetoken.Otherstationsontheringhavetheroleofpassivemonitor.Theprimaryjob of these stations is to detect failure of the active monitor and assume the role of active monitor. A contention-resolution is used to determine which station to take over.

Physical Layer

The Token Ring uses shielded twisted pair of wire to establish point-point links between the adjacent stations. The baseband signaling uses differential Manchester encoding. To overcome the problem of cable break or network failure, which brings the entire network down, one suggested technique, is to use *wiring concentrator* as shown in Fig. 26.

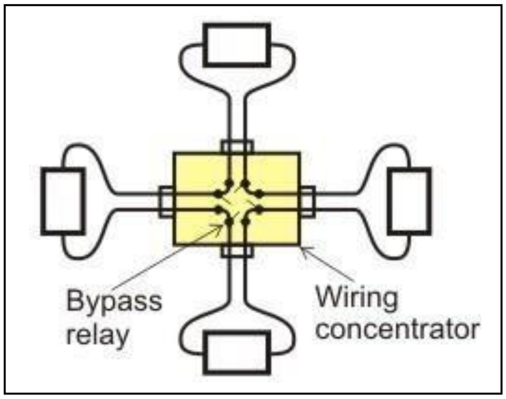


Fig 26 Star Connected Ring topology

It imposes the reliability in an elegant manner. Although logically the network remains as a ring, physically each station is connected to the *wire center* with two twisted pairs for

2- way communication. Inside the wire center, *bypass relays* are used to isolate a broken wire or a faulty station. This Topology is known as Star-Connected Ring**.**

Frame Format

Token Ring and IEEE 802.5 support two basic frame types: tokens and data/command frames. Tokens are 3 bytes in length and consist of a start delimiter, an access control byte, and an end delimiter. Data/command frames vary in size, depending on the size of the Information field. Data frames carry information for upper-layer protocols, while command frames contain control information and have no data for upper-layer protocols.

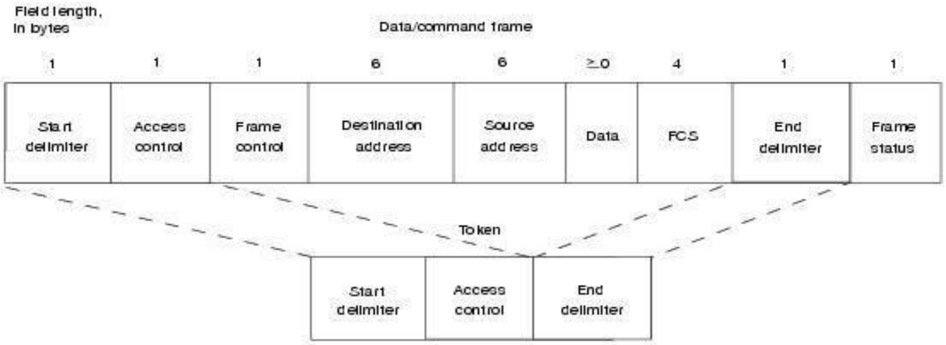


Fig 27 Frame format

Token Ring and IEEE support two basic frame types:

- Tokens
- data/command frames

Tokens are 3 bytes in length and consist of a start delimiter, an access control byte, and an end delimiter. Data/command frames vary in size, depending on the size of the Information field. Data frames carry information for upper-layer protocols, while command frames contain control information and have no data for upper-layer protocols. IEEE 802.5 and Token Ring Specify Tokens and Data/Command Frames

Token Frame Fields

Start	Access	Ending

Token Frame contains three fields, each of which is 1 byte in length:

- **Start delimiter (1 byte)**: Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.
- **Access-control (1 byte)**: Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a token from a data/command frame) and a monitor bit (used by the active monitor to determine whether a frame is circling the ring endlessly).
- **End delimiter (1 byte)**: Signals the end of the token or data/command frame. This field also contains bits to indicate a damaged frame and identify the frame that is the last in a logica lsequence.

Data/Command Frame Fields

Start Delimiter	**Access Contr**	**Frame Contr**	**Destination address**	**Source addre**	**Data**	**Frame check sequence**	**End Delimiter**	**Frame Stat**

Data/command frames have the same three fields as Token Frames, plus several others. The Data/command frame fields are described below:

- **Frame-control byte (1 byte)**—Indicates whether the frame contains data or control information. In control frames, this byte specifies the type of control information.
- **Destination and source addresses (2-6 bytes)**—Consists oftwo6-bytaddress fields that identify the destination and source station addresses.
- **Data (up to 4500 bytes)**—Indicates that the length of field is limited by the ring token holding time, which defines the maximum time a station can hold thetoken.
- **Frame-check sequence (FCS- 4 byte)**—Is filed by the source station with a calculated value dependent on the frame contents. The destination station recalculates the value to determine whether the frame was damaged in transit. If so, the frame is discarded.
- **Frame Status (1 byte)**—This is the terminating field of a command/data frame. The Frame Status field includes the address-recognized indicator and frame-copied indicator.

Token Frame Fields

The three token frame fields descriptions that follow:

- **Start delimiter** - Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.
- **Access-control byte** - Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a token from a data/command frame) and a monitor bit (used by the active monitor to determine whether a frame is circling the ring endlessly).

- **End delimiter** - Signals the end of the token or data/command frame. This field also contains bits to indicate a damaged frame and identify the frame that is the last in a logical sequence.

Data/Command Frame Fields

Data/command frames have the same three fields as Token Frames, plus several others. The Data/command frame fields are described in the following summaries:

- **Start delimiter** - Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.
- **Access-control byte** - Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a tokenfromadata/commandframe)andamonitorbit(usedbytheactivemonitortodetermine whether a frame is circling the ring endlessly).

- **Frame-control bytes** - Indicates whether the frame contains data or control information. In control frames, this byte specifies the type of control information.
- **Destination and source addresses** - Consists of two 6-byte address fields that identify the destination and source station addresses.
- **Data** - Indicates that the length of field is limited by the ring token holding time, which defines the maximum time a station can hold the token.
- **Frame-check sequence (FCS)** - Is filed by the source station with a calculated value dependent on the frame contents. The destination station recalculates the value to determine whether the frame was damaged in transit. If so, the frame is discarded.
- **End Delimiter** - Signals the end of the token or data/command frame. The end delimiter also contains bits to indicate a damaged frame and identify the frame that is the last in a logical sequence.
- **Frame Status** - Is a 1-byte field terminating a command/data frame. The Frame Status field

includes the address-recognized indicator and frame-copied indicator.

7. Token Bus (IEEE 802.4)

Token BUS: A Brief History

Although Ethernet was widely used in the offices, but people interested in factory automation did not like it because of the probabilistic MAC layer protocol. They wanted a protocol which can support priorities and has predictable delay. These people liked the conceptual idea of Token Ring network but did not like its physical implementation as a break in the ring cable could bring the whole network down and ring is a poor fit to their linear assembly lines. Thus a new standard, known as Token bus, was developed, having therobustnessoftheBustopology,buttheknownworst-casebehaviorofaring.Here stations are logically connected as a ring but physically on a Bus and follows the collision-free token passing medium access control protocol. So the motivation behind token bus protocol can be summarized as:

- The probabilistic nature of CSMA/ CD leads to uncertainty about the delivery time; which created the need for a different protocol
- The token ring, on the hand, is very vulnerable to failure.
- Token bus provides deterministic delivery time, which is necessary for real time traffic.
- Token bus is also less vulnerable compared to token ring.

Functions of a Token Bus

It is the technique in which the station on bus or tree forms a logical ring, that is the stations are assigned positions in an ordered sequence, with the last number of the sequence followed by the first one as shown in Figure. Each station knows the identity of the station following it and preceding it.

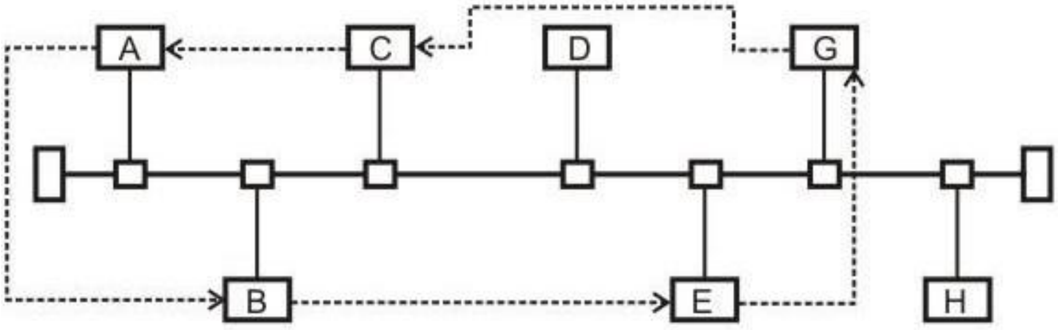


Fig 28 Token Bus topology

A control packet known as a *Token* regulates the right to access. When a station receives the token, it is granted control to the media for a specified time, during which it may transmit one or more packets and may poll stations and receive responses when the station is done, or if its time has expired then it passes token to next station in logical sequence. Hence, steady phase consists of alternate phases of token passing and data transfer.

The MAC sublayer consists of four major functions: the interface machine (IFM), the access control machine (ACM), the receiver machine (RxM) and the transmit machine (TxM).

IFM interfaces with the LLC sublayer. The LLC sublayer frames are passed on to the ACM by the IFM and if the received frame is also an LLC type, it is passed from RxM component to the LLC sublayer. IFM also provides quality of service.

The **ACM** is the heart of the system. It determines when to place a frame on the bus, and responsible for the maintenance of the logical ring including the error detection and fault recovery. It also cooperates with other stations ACM's to control the access to the shared bus, controls the admission of new stations and attempts recovery from faults and failures.

The responsibility of a **TxM** is to transmit frame to physical layer. It accepts the frame from the ACM and builds a MAC protocol data unit (PDU) as per the format.

The **RxM** accepts data from the physical layer and identifies a full frame by detecting the SD and ED (start and end delimiter). It also checks the FCS field to validate an error- free transmission.

Frame Form

The frame format of the Token Bus is shown in Figure. Most of the fields are sameas Token Ring.

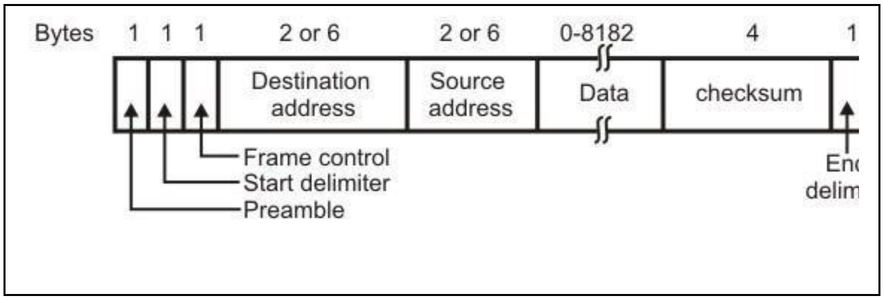


Fig 29 Token Bus frame format

Logical ring maintenance

The MAC performs the following functions as part of its maintenance role of the ring.

Addition to the Ring: Non-participating stations must periodically be granted the opportunity to insert themselves into the ring. Each node in the ring periodically grants an opportunity for new nodes to enter the ring while holding the token. The node issues a solicit–successor–1 packet, inviting nodes with an address between itself and the nextnode in logical sequence to request entrance. The transmitting node then waits for a period of time equal to one response window or slot time (twice the end-to-end propagation delay of the medium). If there is no request, the token holder sets its successor node to be the requesting node and transmits the token to it; the requester sets the linkages accordingly and proceeds.

If more than one node requests, to enter the ring, the token holder will detect a garbled transmission. The conflict is resolved by *addressed based contention scheme;* the token holder transmits a resolved contention packet and waits for four response windows. Each requester can transmit in one of these windows, based on the first two bits of its address. If requester hears anything before its windows comes up, it refrains from requesting entrance. If a token holder receives a valid response, then it can proceed, otherwise it tries again and only those nodes that request the first time are allowed to request this time, based on the second pair of bits in their address. This process continues until a valid request is received or no request is received, or a maximum retry count is reached. In latter cases, the token holder passes the token to logical successor in the ring.

Deletion from Ring: A station can voluntarily remove itself from the ring by splicing together its predecessor and successor. The node which wants to be deleted from the ring waits until token comes to it, then it sends a set successor packet to its predecessor, instructing it to splice to its successor.

Fault Management: Errors like duplicate address or broken ring can occur. A suitable management scheme should be implemented for smooth functioning. It is done by the token-holder first, while holding the token, node may hear a packet, indicating that another node has the token. In this case, it immediately drops the token by reverting to listener mode, and the number of token holders drops immediately from one to zero. Upon completion of its turn, it immediately issues a data or token packet. The sequence of steps are as follows:

i. After sending the token, the token issuer will listen for one slot time to make sure that its predecessor is active.
ii. If the issuer does not hear a valid packet, it reissues the token to the same successor one more time.
iii. After two failures, the issuer assumes that its successor has failed and issues a "who-follows" packet, asking for the identity of the node that follows the failed node. The issuer should get back a set successor packet from the second node down the time. If so, the issuer adjusts its linkage and issues a token .
iv. If the issuing node gets a response to its "who-follows" packet, it tries again.
v. If the "who-follows" tactic fails, the node issues a solicit-successor-2 packet with full address range (i.e. every node is invited to respond). If this packet works then the ring is established and procedure continues.
vi. If two attempts in step (v) fail, it assumes that a catastrophe has happened; perhaps the node receiver has failed. In any case, the node ceases the activity and listen the bus.

Ring Initialization: Ring is to be initialized by starting the token passing. This is necessary when the network is being setup or when ring is broken down. Some decentralized algorithms should take care of, who starts first, who starts second, etc. it occurs when one or more stations detects a lack of bus activity lasting longer than a specific time. The token may get lost. This can occur on a number of occasions. For example, when network has been just powered up, or a token holding station fails. Once its time out expires, a node will issue a claim token packet. Contending clients are removed in a similar fashion to the response window process.

Relative comparison of the three standards

A comparison of the three standards for different functions is shown in Table and results of the analysis of the performance of the three standards are summarized below:

The CSMA/CD protocol shows strong dependence on the parameter 'a', which is the ratio of the propagation time to the transmission time. It offers shortest delay under light load and it is most sensitive under heavy load conditions.
Token ring is least sensitive to different load conditions and different packet sizes.
Token bus is highly efficient under light load conditions

8. SMDS

Switched Multimegabit Data Service (SMDS) is a packet-switched datagram service designed for very high-speed wide-area data communications. SMDS offers data throughputs that will initially be in the 1- to 34-Mbps range.

Ethernet

SMDS data units are capable of containing up to 9,188 octets (bytes) of user information. SMDS is therefore capable of encapsulating entire IEEE 802.3, IEEE 802.4, IEEE 802.5, and FDDI frames. The large packet size is consistent with the high-performance objectives of the service.

Addressing

Like other datagram protocols, SMDS data units carry both a source and a destination address. The recipient of a data unit can use the source address to return data to the sender and for functions such as address resolution (discovering the mapping between higher-layer addresses and SMDS addresses). SMDS addresses are 10-digit addresses that resemble conventional telephone numbers. In addition, SMDS supports group addresses that allow a single data unit to be sent and then delivered by the network to multiple recipients. Group addressing is analogous to multicasting on local-area networks (LANs) and is a valuable feature in internetworking applications where it is widely used for routing, address resolution, and dynamic discovery of network resources.

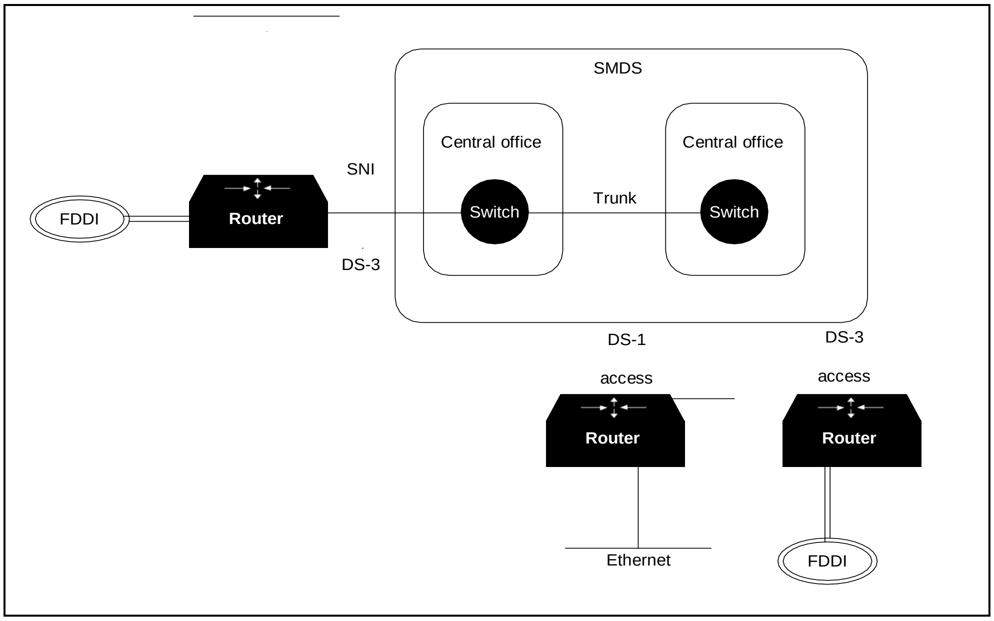


Fig 30 SMDS Operation

SMDS offers several other addressing features. Source addresses are validated by the network to ensure that the address in question is legitimately assigned to the SNI from which it originated. Thus, users are protected against address spoofing—that is, a sender pretending to be another user. Source and destination address screening is also possible. Source address screening acts on addresses as data units are leaving the network, while destination address screening acts on addresses as data units are entering the network. If the address is disallowed, the data unit is not delivered. With address screening, a subscriber can establish a private virtual network that excludes unwanted traffic.

SMDS data units are capable of containing up to 9,188 octets (bytes) of user information. SMDS is therefore capable of encapsulating entire IEEE 802.3, IEEE 802.4, IEEE 802.5, and FDDI frames. The large packet size is consistent with the high-performance objectives of the service.

Access Classes

To accommodate a range of traffic requirements and equipment capabilities, SMDS supports a variety of access classes. Different access classes determine the various maximum sustained information transfer rates as well as the degree of burstiness allowed when sending packets into the SMDS network.

On DS-3-rate interfaces, access classes are implemented through credit management algorithms, which track credit balances for each customer interface. Credit is allocated on a periodic basis, up to some maximum. Then, the credit balance is decremented as packets are sent to the network.

The operation of the credit management scheme essentially constrains the customer's equipment to some sustained or average rate of data transfer. This average rate of transfer is less than the full information carrying bandwidth of the DS-3 access facility. Five access classes, corresponding to sustained information rates of 4, 10, 16, 25, and 34 Mbps, are supported for DS-3 access interface. The credit management scheme is not applied to DS-1-rate access interfaces.

SMDS Interface Protocol (SIP)

Access to the SMDS network is accomplished via SIP. The SIP is based on the DQDB protocol specified by the IEEE 802.6 MAN standard. The DQDB protocol defines a Media Access Control (MAC) protocol that allows many systems to interconnect via two unidirectional logical buses.

As designed by IEEE 802.6, the DQDB standard can be used to construct private, fiber-based MANs supporting a variety of applications including data, voice, and video. This protocol was chosen as the basis for SIP because it was an open standard, could support all the SMDS service features, was designed for compatibility with carrier transmission standards, and is aligned with emerging standards for Broadband ISDN (BISDN). As BISDN technology matures and is deployed, the carriers intend to support not only SMDS but broadband video and voice services as well.

To interface to SMDS networks, only the connectionless data portion of the IEEE 802.6 protocol is needed. Therefore, SIP does not define voice or video application support.

When used to gain access to an SMDS network, operation of the DQDB protocol across the SNI results in an access DQDB. The term access DQDB distinguishes operation of DQDB across the SNI from operation of DQDB in any other environment (such as inside the SMDS network). A switch in the SMDS network operates as one station on an access DQDB, while customer equipment operates as one or more stations on the access DQDB.

Because the DQDB protocol was designed to support a variety of data and nondata applications and because it is a shared medium access control protocol, it is relatively complex. It has two parts:

- The protocol syntax
- The distributed queuing algorithm that constitutes the shared medium access control

9. Fiber-Distributed Data Interface (FDDI)

Introduction

The IEEE 802.3 and 802.5 LANs, discussed in the previous sections, having data transfer rate in the range of 10 Mb/s to 16 Mb/s have served the purpose very well for many years. But with the availability of powerful computers at a low cost and emergence of new applications, particularly based on multimedia, there is a growing demand for higher network bandwidth. The combined effect of the growth in the number of users and increasing bandwidth requirement per user has led to the development of High Speed. LANs with data transfer rate of 100 Mb/s or more.

The high speed LANs that have emerged can be broadly categorized into three types *based on token passing, successors of Ethernet* and *based on switching technology*. In the first category we have *FDDI* and its variations, and high-speed token ring. In the second category we have the *fast Ethernet* and *Gigabit Ethernet*. In the third category we have *ATM*, *fiber channel* and the *Ether switches*. In this lesson we shall discuss details of FDDI – the token ring based high speed LAN.

FDDI

Fiber Distributed Data Interface (FDDI), developed by American National Standards Institute (ANSI) is a token passing ring network that operates at 100 Mb/s on optical fiber-medium. Its medium access control approach has close similarity with the IEEE
802.5 standard, but certain features have been added to it for higher reliability and better performance. Key features of FDDI are outlined in this section.

The FDDI standard divides transmission functions into 4 protocols: physical medium dependent (PMD), Physical (PHY), media access control(MAC) and Logical link control(LLC) as shown in Fig. These protocols correspond to the physical and data link layer of OSI reference model. Apart from these four protocols, one more protocol which span across both data link and physical layer (if considered of OSI), used for the station management.

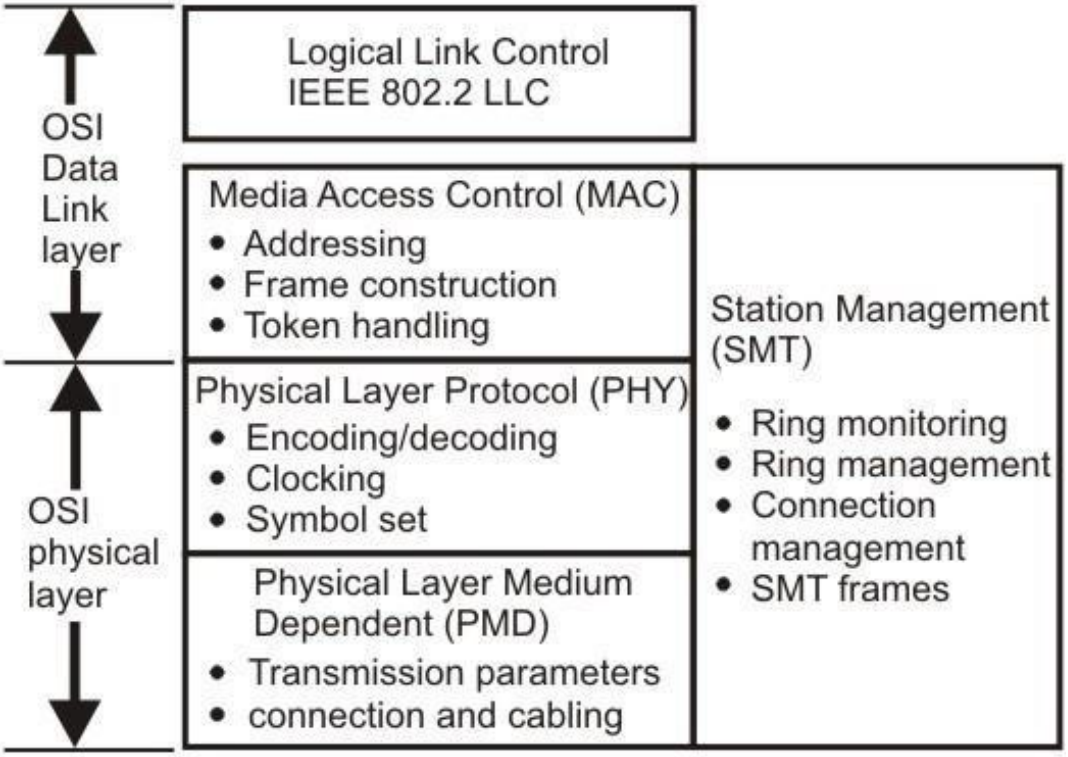


Fig 31 FDDI protocols

FDDI networks offered transmission speeds of 100 Mbps, which initially made them quite popular for high-speed networking. With the advent of 100-Mbps Ethernet, which is
cheaper and easier to administer, FDDI has waned in popularity.

FDDI (Fiber-Distributed Data Interface) is a standard for data transmission on fiber optic lines in that can extend in range up to 200 km (124 miles). The FDDI protocol is based on the token ring protocol. In addition to being large geographically, an FDDI local area network can support thousands of users.

An FDDI network contains two token rings, one for possible backup in case the primary ring fails. The primary ring offers up to 100 Mbps capacity. If the secondary ring is not needed for backup, it can also carry data, extending capacity to 200 Mbps. The single ring can extend the maximum distance; a dual ring can extend 100 km (62miles).

FDDI is a product of American National Standards Committee X3-T9 and conforms to the open system interconnect (OSI) model of functional layering. It can be used to interconnect LANs using other protocols.

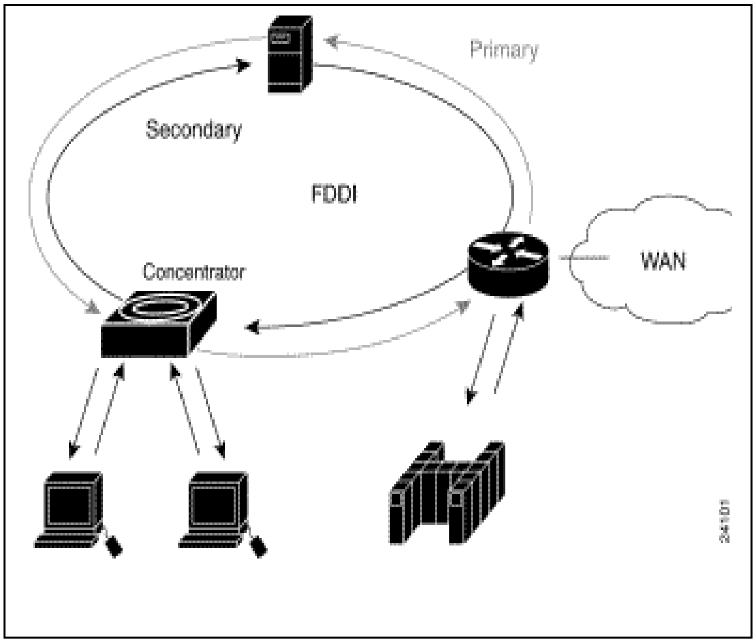


Fig 31 FDDI Operation

Function of FDDI

- The Fiber Distributed Data Interface (FDDI) specifies a 100-Mbps token-passing, dual-ring LAN using fiber-optic cable. FDDI is frequently used as high-speed backbone technology because of its support for high bandwidth and greater distances than copper.
- FDDI uses a dual-ring architecture with traffic on each ring flowing in opposite directions (called counter- rotating).
- The dual-rings consist of a primary and a secondary ring.
- During normal operation, the primary ring is used for data transmission, and the secondary

ring remains idle.

- The primary purpose of the dual rings, as will be discussed in detail later in this chapter, is to provide superior reliability and robustness. Figure 1 shows the counter-rotating primary and secondary FDDI rings.

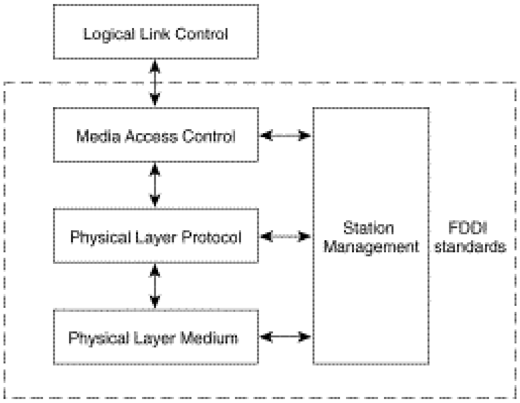


Fig 32 Layers of FDDI

FDDI Station-Attachment Types

One of the unique characteristics of FDDI is that multiple ways actually exist by which to connect FDDI devices. FDDI defines three types of devices: single-attachment station (SAS), dual-attachment station (DAS), and a concentrator.

An SAS attaches to only one ring (the primary) through a concentrator. One of the primary advantages of connecting devices with SAS attachments is that the devices will not have any effect on the FDDI ring if they are disconnected or powered off. Concentrators will be discussed in more detail in the following discussion. Each FDDI DAS has two ports, designated A and B. These ports connect the DAS to the dual FDDI ring. Therefore, each port provides a connection for both the primary and the secondary ring. As you will see in the next section, devices using DAS connections will affect the ring if they are disconnected or powered off. Figure 3 shows FDDI DAS A and B ports with attachments to the primary and secondary rings. An FDDI concentrator (also called a dual-attachment concentrator [DAC]) is the building block of an FDDI network. It attaches directly to both the primary and secondary rings and ensures that the failure or power-down of any SAS does not bring down the ring.

FDDI Physical layer specification

FDDI uses 4B/5B code for block coding. The 5-bit code is selected such that it has no more than one leading zero and no more than two trailing zeros and more than three consecutive 0's do not occur. Table 5.5.2 shows the encoded sequence for all the 4-bit data sequences. The This is normally line coded with NRZ-I.

Topology

The basic topology for FDDI is *dual counter rotating rings:* one transmitting clockwise and the other transmitting counter clockwise as illustrated in the Fig. One is known as *primary ring* and the other *secondary ring*. Although theoretically both the rings can be used to achieve a data transfer rate of 200 Mb/s, the standard recommends the use of the primary ring for data transmission and secondary ring as a backup.

In case of failure of a node or a fiber link, the ring is restored by wrapping the primary ring to the secondary ring. The redundancy in the ring design provides a degree of fault tolerance, not found in other network standards. Further improvement in reliability and availability can be achieved by using *dual ring* of trees and *dual homing* mechanism.

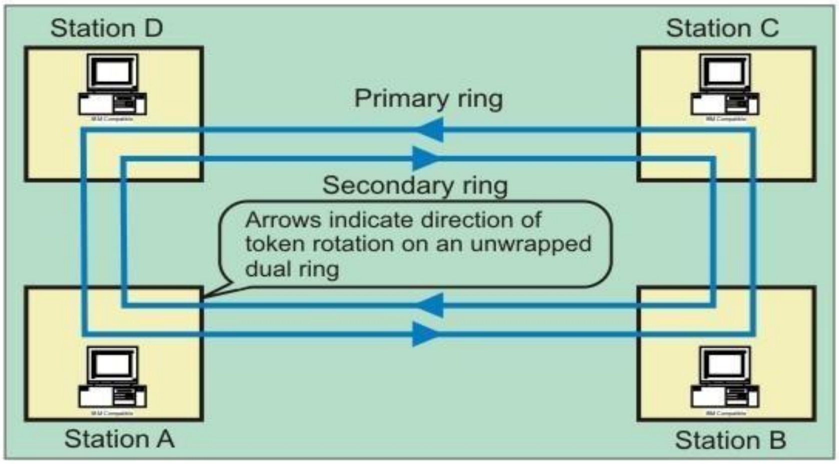

Station D
Station C
Primary ring
Secondary ring
Arrows indicate direction of token rotation on an unwrapped dual ring
Station A
Station B

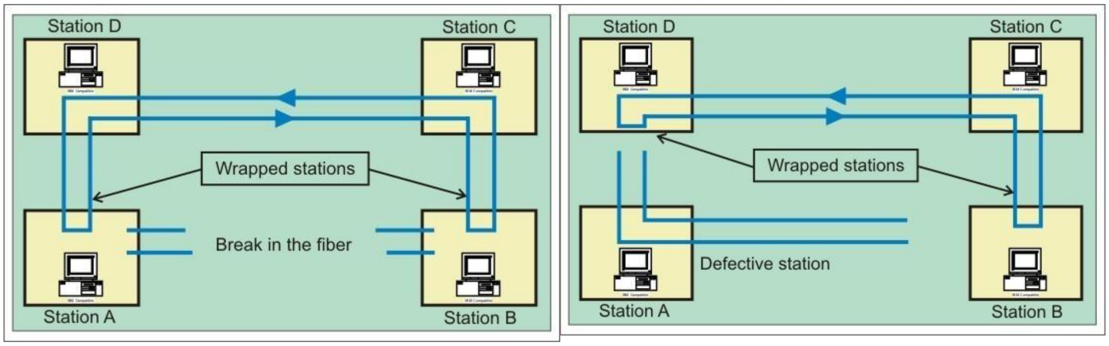
Fig 33 FDDI dual counter-rotating ring topology

Fault Tolerance

FDDI provides a number of fault-tolerant features. In particular, FDDI's dual-ring environment, the implementation of the optical bypass switch, and dual-homing support make FDDI a resilient media technology.

Dual Ring

FDDI's primary fault-tolerant feature is the *dual ring*. If a station on the dual ring fails or is powered down, or if the cable is damaged, the dual ring is automatically wrapped (doubled back onto itself) into a single ring. When the ring is wrapped, the dual-ring topology becomes a single-ring topology. Data continues to be transmitted on the FDDI ring without performance impact during the wrap condition. Figure (a) and Figure (b) illustrate the effect of a ring wrapping in FDDI.



(b)

Fig 34 FDDI ring with a (a) broken link, (b) defective station

When a cable failure occurs, devices on either side of the cable fault wrap. Network operation continues for all stations. When a single station fails ,devices on either side of the failed (or powered-down) station wrap, forming a single ring. Network operation continues for the remaining stations on the ring. It should be noted that FDDI truly provides fault tolerance against a single failure only. When two or more failures occur, the FDDI ring segments into two or more independent rings that are incapable of communicating with eachother.

Optical Bypass Switch

An optical bypass switch provides continuous dual-ring operation if a device on the dual ring fails. This is used both to prevent ring segmentation and to eliminate failed stations from the ring. The optical bypass switch performs this function using optical mirrors that pass light from the ring directly to the DAS (dual-attachment station) device during normal operation. If a failure of the DAS device occurs, such as a power-off, the optical bypass switch will pass the light through itself by using internal mirrors and thereby will maintain the ring's integrity.

Failed Station

The benefit of this capability is that the ring will not enter a wrapped condition incase of a device failure. A somewhat similar technique has been discussed in Token ring section (Star Connected Ring- where relays are used to bypass the faulty node). Figure shows the functionality of an optical bypass switch in an FDDI network. When using the OB, you will notice a tremendous digression of your network as the packets are sent through the OB unit.

Dual Homing: Critical devices, such as routers or mainframe hosts, can use a fault- tolerant technique called *dual homing* to provide additional redundancy and to help guarantee operation. In dual-homing situations, the critical device is attached to two concentrators.

Frame Format

Each Frame is preceded by a preamble (16 idle symbols-1111), for a total of 64 bits, to initialize clock synchronization with the receiver.

PA:	Preamble
SD:	Starting Delimiter
FC:	Frame Control
DA :	Destination Address
SA:	Source Address
FCS:	Frame Check Sequence
ED:	Ending Delimiter
FS:	Frame Status

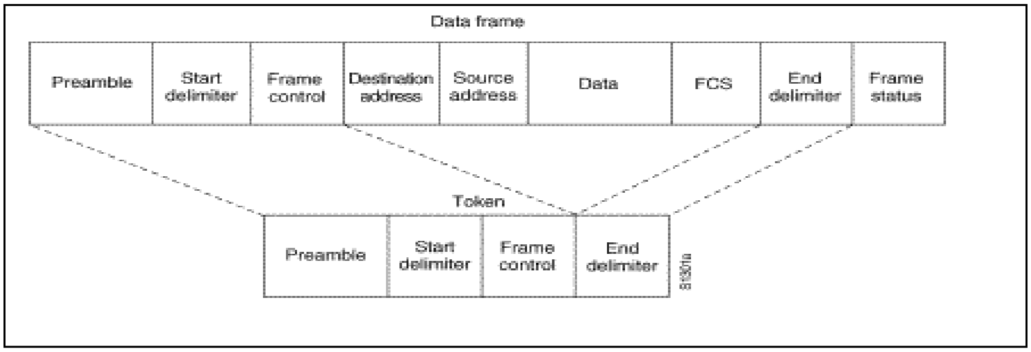


Fig 35 Frame format for the FDDI

Let us have a look at the various fields:
SD: The first byte, after the preamble, of the field is the frame's starting flag. As in Token ring these bits are replaced in physical layer by the control codes.

FC: it identifies the frame type i.e. token or a data frame.
Address: the next 2 fields are destination and source addresses. Each address consists of 2-6 bytes.

Data: Each data frame carries up to 4500 bytes.

FCS: FDDI uses the standard IEEE four-byte cyclic redundancy check.
ED: this field consists of half a byte in data frame or a full byte in token frame. This represents end of the Token.

FS: FDDI FS field is similar to that of Token Ring. It is included only in data/Command frame and consists of one and a half bytes.

Media Access Control

The FDDI media access control protocol is responsible for the following services.

(i) Fair and equal access to the ring by using a *timed token protocol*. To transmit on the ring, a station must first acquire the token. A station holds the token until it has transmitted all of its frames or until the transmission time for the appropriate service is over. Synchronous traffic is given a guaranteed bandwidth by ensuring that token rotation time does not exceed a preset value. FDDI implements these using three timers, *Token holding Timer* (THT), which determines how long a station may continue once it has captured a token. *Token Rotation Timer* (TRT) is reset every time a token is seen. When timer expires, it indicates that the token is lost and recovery is started. The Valid Transmission Timer (VTT) is used to time out and recover from some transmit ring errors.

(ii) Construction of frames and tokens are done as per the format shown in Figure 5.5.5. The frame status (FS) byte is set by the destination and checked by the source station, which removes its frame from the ring and generates another token.

(iii) Transmitting, receiving, repeating and stripping frames and tokens from the ring, unlike IEEE 802.5, is possible for several frames on the ring simultaneously. Thus a station will transmit a token immediately after completion of its frame transmission. A station further

(iv) down the ring is allowed to insert its own frame. This improves the potential throughput of the system. When the frame returns to the sending station, that station removes the frame from the ring by a process called *stripping*.

(v) It also does *ring initialization*, *fault isolation* and error detection as we have discussed for IEEE802.5.

10. DQDB-IEEE 802.6

The IEEE Standard 802.6 *(*Distributed Queue Dual Bus *(DQDB) Sub network of a* Metropolitan Area Network (MAN)) permits sub network reconfiguration, usually without loss of communication ability, whenever there are bus faults. The Configuration Control Protocol (CCP) is the protocol which enables this to occur.

IEEE 802.6 is a standard governed by the ANSI for Metropolitan Area Networks (MAN). It is an improvement of an older standard (also created by ANSI) which used the Fiber distributed data interface (FDDI) network structure.The FDDI-based standard failed due to its expensive implementation and lack of compatibility with current LAN standards. The IEEE 802.6 standard uses the Distributed Queue Dual Bus (DQDB) network form. This form supports 150 Mbit/s transfer rates. It consists of two unconnected unidirectional buses. DQDB is rated for a maximum of 160 km before significant signal degradation over fiber optic cable with an optical wavelength of1310 nm. This standard has also failed, mostly due to the same reasons that the FDDI standard failed. Most MANs now use Synchronous Optical Network (SONET) or Asynchronous Transfer Mode (ATM) network designs, with recent designs using native Ethernet or MPLS.

The DQDB Physical Layer

The Head of Bus (HOB)s act a slot generators so that the bus is never quiet.Nodes are located logically adjacent to the bus and are promiscuous readers. They read all slots that come off the bus but may not necessarily alter any of the data. Nodes may be passive readers or, in an active system, they may act as repeaters so as to forestall attenuation. If Node 2 wishes to send data in the direction of Node n then it will use Bus A. This implies that it must first reserve a slot by placing a request on Bus B.If Node 2 wishes to send data in the direction of Node 1 it must first reserve a slot using Bus A and then send the data on Bus B.

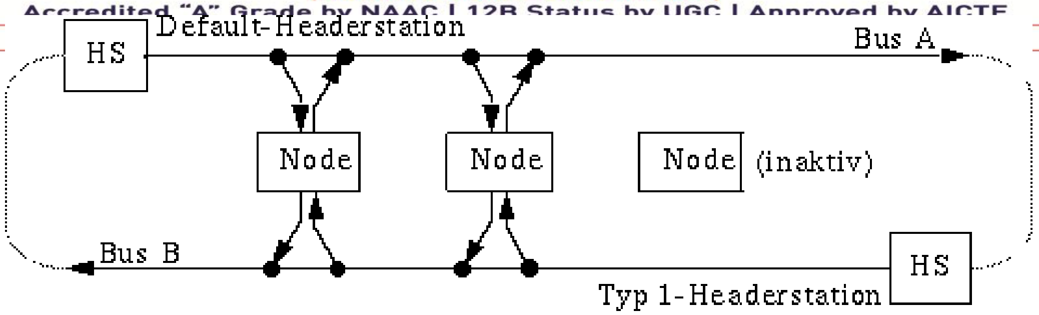


Fig 36 DQDB Operation

DQDB Operation

- The DQDB configuration is independent of the number of nodes and of the distances involved making DQDB ideal for high-speed transmissions
- DQDB uses 53-byte packets (52 data bytes and one access control byte) for transmissions called slots.
    - Slots from different nodes are intermingled in the network traffic.

- The head node (the first node connected to the external fiber) is responsible for creating empty slots and sending these down the line to the other nodes to use.
- The down line nodes indicate how many slots are needed using the secondary bus to the head node which then creates empty slots and sends these down the line.
    - As the slots move down the line, they are taken by the nodes that have requested them.

Switching

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). These methods, however, are impractical and wasteful when applied to very large networks. The number and length of the links require toomuch infrastructure to be cost-efficient, and the majority of those links would be idle most of the time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment.

A better solution is switching. A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure shows a switchednetwork.

The end systems (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

Taxonomy of switched networks

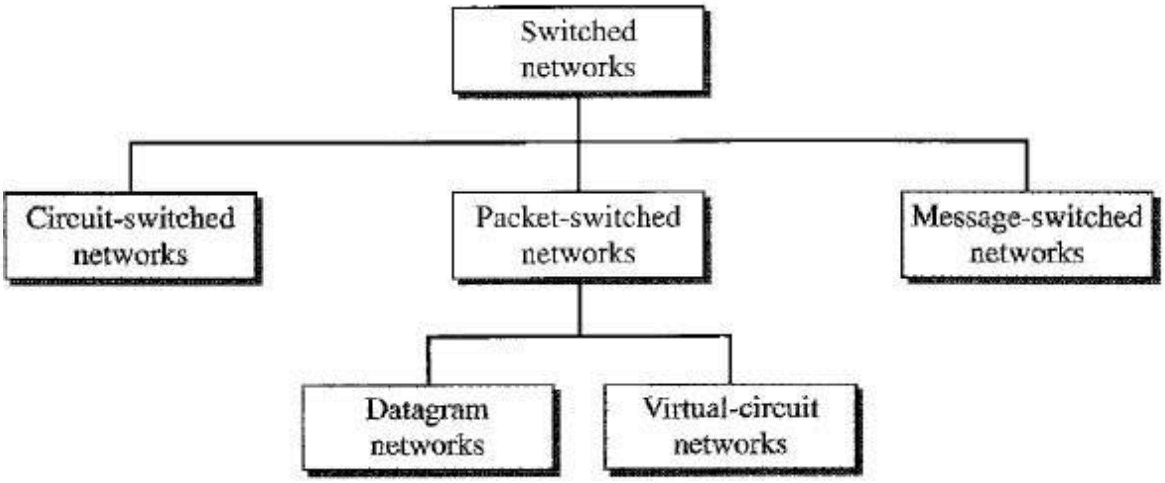


Fig 37 Types of switched Networks

11. CIRCUIT-SWITCHED NETWORKS

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link. Each link is normally divided into *n* channels by using FDM or TDM. Figure shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n (n* is 3 in the figure) channels by using

FDM or TDM.

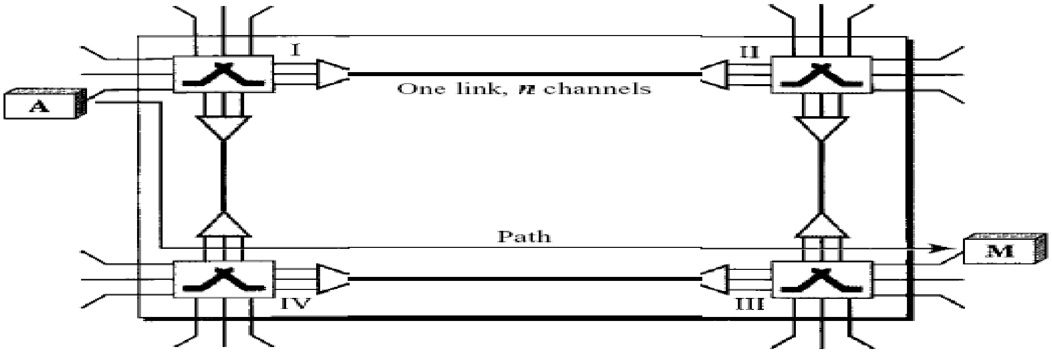


Fig 38 A trivial circuit-switched network

Three Phases

The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

***Setup Phase**:*

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches. For example, in Figure, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time. In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established. Note that end-to-end addressing is required for creating a connection between the two end systems. These can be, for example, the addresses of the computers assigned by the administrator in a TDM network, or telephone numbers in an FDM network.

***Data Transfer Phase**:*

After the establishment of the dedicated circuit (channels), the two parties can

Transfer data.

Teardown Phase:

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

Efficiency:

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation. However, in computer networks, a computer can be connected to another computer even if there is no activity for a long time. In this case, allowing resources to be dedicated means that other connections are deprived.

Delay

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. As Figure shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit.

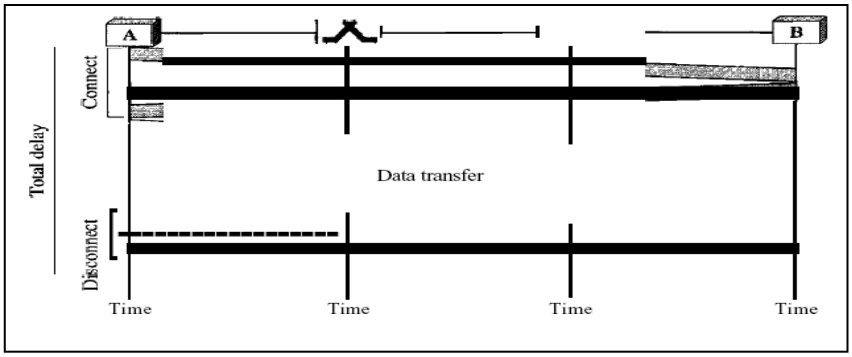


Fig 39 Delay in a circuit-switched network

The delay caused by the setup is the sum of four parts: the propagation time of the source computer request (slope of the first gray box), the request signal transfer time (height of the first gray box), the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and the signal transfer time of the acknowledgment (height of the second gray box). The delay due to data transfer is the sum of two parts: the propagation time (slope of the colored box) and data transfer time (height of the colored box), which can be very long. The third box shows the time needed to tear down the circuit. We have shown the case in which the receiver requests disconnection, which creates the maximum delay.

DATAGRAM NETWORKS

In a datagram network, each packet is treated independently of all others. Even if a packet is part of a multi packet transmission, the network treats it as though it existed alone. Packets in this approach are referred to as datagrams.

Datagram switching is normally done at the network layer. We briefly discuss datagram networks here as a comparison with circuit-switched and virtual-circuit switched networks Figure shows how the datagram approach is used to deliver four packets from station A to station

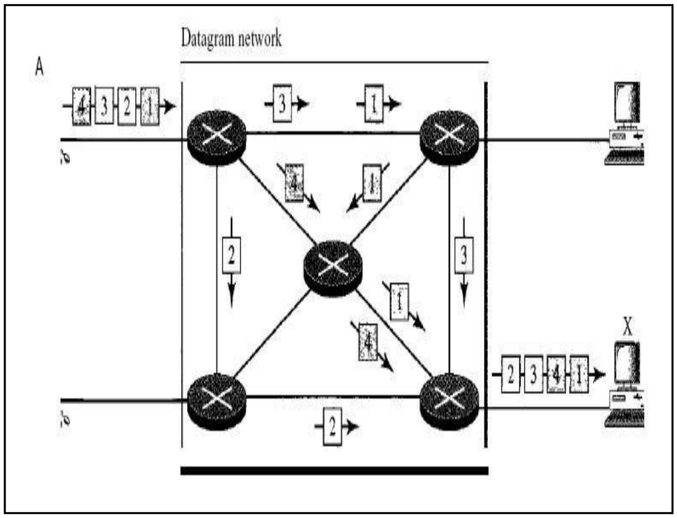


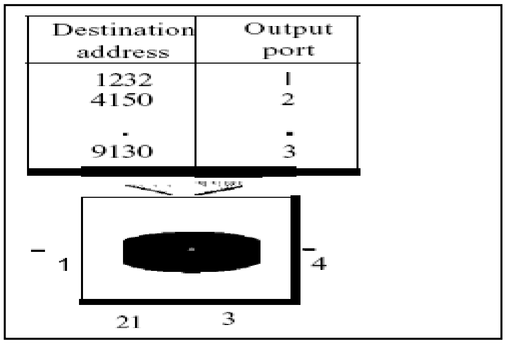
Fig 40 Datagram network with four switches (routers)

The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure. In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X. This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between the packets. Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upper- layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application.

The datagram networks are sometimes referred to as connectionless networks. The term *connectionless* here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. This is different from the table of a circuit switched network in which each entry is created when the setup phase is completed and deleted when the teardown phase is over. Figure shows the routing table for a switch.



Destination address	Output port
1232	1
4150	2
.	.
9130	3

Fig 41 Routing table in a datagram network

Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded. This address, unlike the address in a virtual-circuit-switched network, remains the same during the entire journey of the packet.

Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these minutes for other packets from other sources.

Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message.

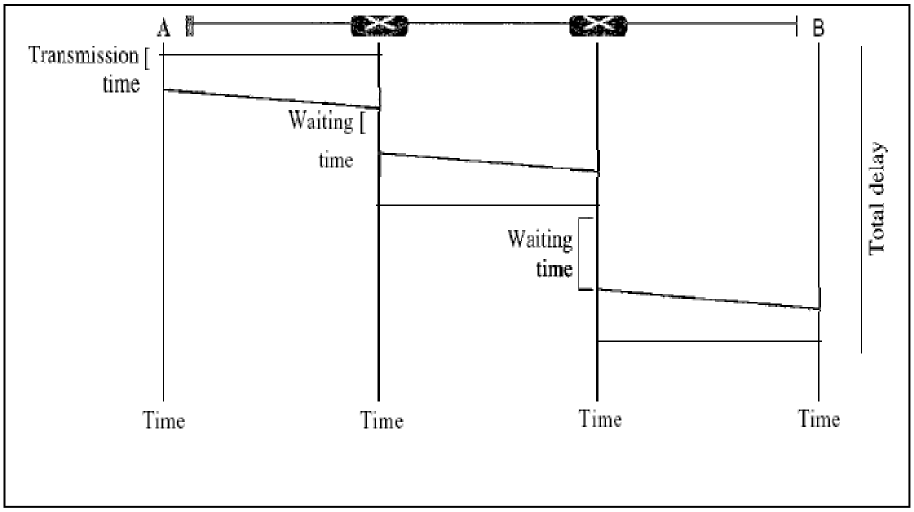


Fig 42 Delay in a datagram network

The packet travels through two switches. There are three transmission times *(3T),* three propagation delays (slopes 3't of the lines), and two waiting times (WI + *w2)'* We ignore the processing time in each switch. The total delay is
Total delay =*3T* + 3t + WI + W2

VIRTUAL-CIRCUIT NETWORKS:

A virtual-circuit network is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what should be the next switch and the channel on which the packet is being canied), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual- circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data link layer, while a circuit- switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future. Figure is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

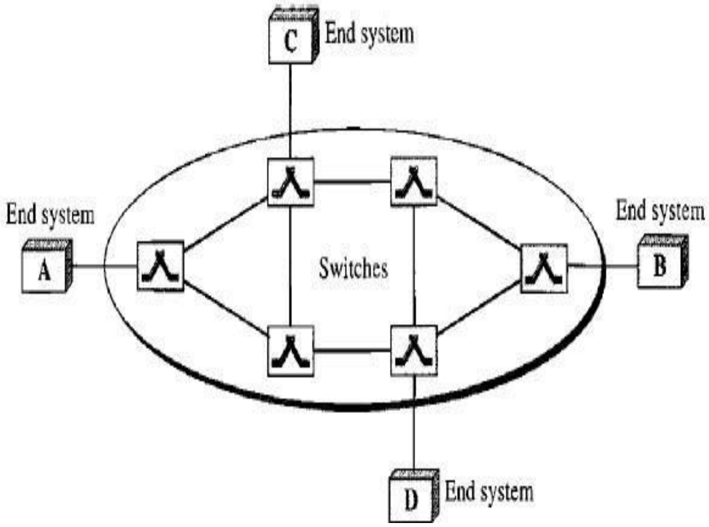


Fig 43 Virtual-circuit network Addressing

In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

Global Addressing*:* A source or a destination needs to have a global address-an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier*:* The identifier that is actually used for data transfer is called the virtual- circuit identifier (Vel). A vel, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCl.

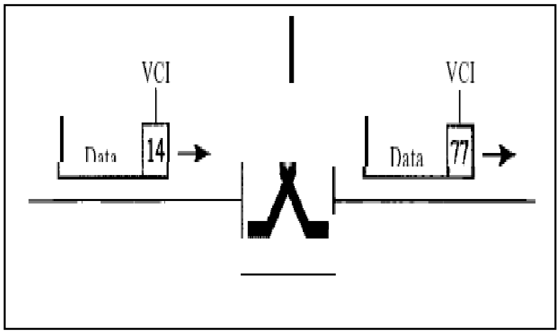


Fig 45*Virtual-circuit identifier*

Three Phases

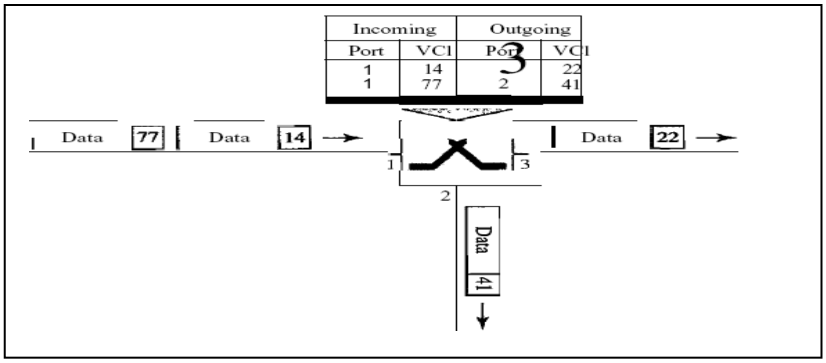
As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown. In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection. In the teardown phase, the source and destination inform the switches to delete the corresponding entry. Data transfer occurs between these two phases. We first discuss the data transfer phase, which is more straightforward; we then talk about the setup and teardown phases.

Data Transfer Phase

To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits. And also shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3. Figure shows how a frame from source A reaches destination B and how its VCI changes during the trip. Each switch changes the VCI and routes the frame. The data transfer phase is active until the source sends all its frames to the destination. The procedure at the switch is the same for each frame of a message. The process creates a virtual circuit, not a real circuit, between the source and destination.

Setup Phase

In the setup phase, a switch creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to B. Two steps are required: the setup request and the acknowledgment.



Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
1	77	2	41

Fig 46 Switch and tables in a virtual-circuit network

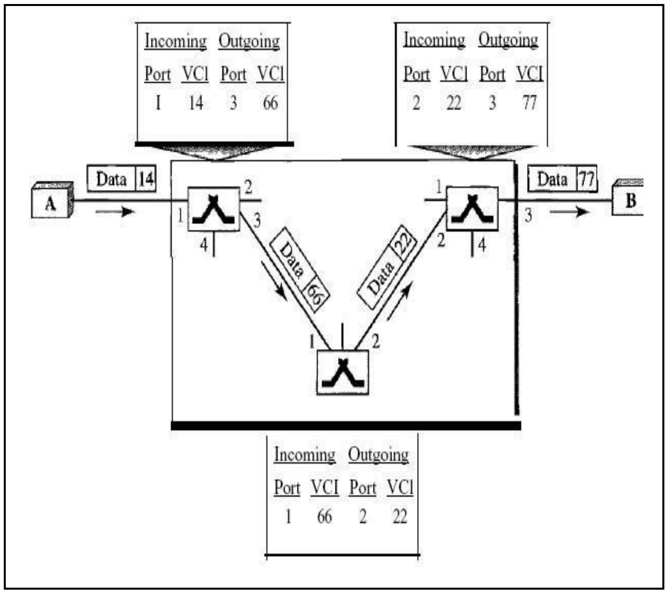


Fig 47 ***Source-to-destination data transfer in a virtual-circuit network***

Setup Request

A setup request frame is sent from the source to the destination. Figure 4 shows the process.

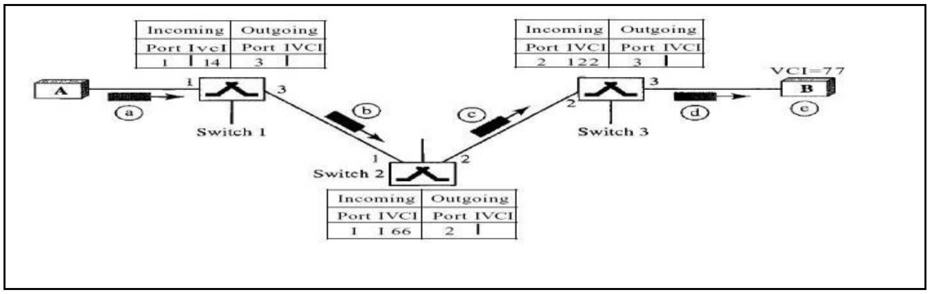


Fig 48 Setup request in a virtual-circuit network

a. Source A sends a setup frame to switch1.

b. Switch 1 receives the setup request frame. It knows that a frame going from A to B

c. goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; it has a routing table which is different from the switching table. For the moment, assume that it knows the output port. The switch creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the outgoing port (3). It does not yet know the outgoing VCI, which will be foundduring
the acknowledgment step. The switch then forwards the frame through port 3 to switch 2.

d. Switch 2 receives the setup request frame. The same events happen here as at switch 1; three columns of the table are completed: in this case, incoming port (l), incoming VCI (66), and outgoing port(2).

e. Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port(3).

f. Destination B receives the setup frame, and if it is ready to receive frames from A, it assigns a VCI to the incoming frames that come from A, in this case 77. This VCI lets the destination know that the frames come from A, and not other sources.

Acknowledgment A special frame, called the acknowledgment frame, completes the entries in the switching tables. The destination sends an acknowledgment to switch 3. The acknowledgment carries the global source and destination addresses so the switch knows which entry in the table is to be completed. TheframealsocarriesVCI77,chosen by the destination as the incoming VCI for frames from A. Switch 3 uses this VCI to complete the outgoing VCI column for this entry. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.

a. Switch 3 sends an acknowledgment to switch 2 that contains its incoming VCI in the table, chosen in the previous step. Switch 2 uses this as the outgoing VCI in the table.

b. Switch 2 sends an acknowledgment to switch 1 that contains its incoming VCI in the table, chosen in the previous step. Switch 1 uses this as the outgoing VCI in the table.

c. Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.

d. The source uses this as the outgoing VCI for the data frames to be sent to destination B.

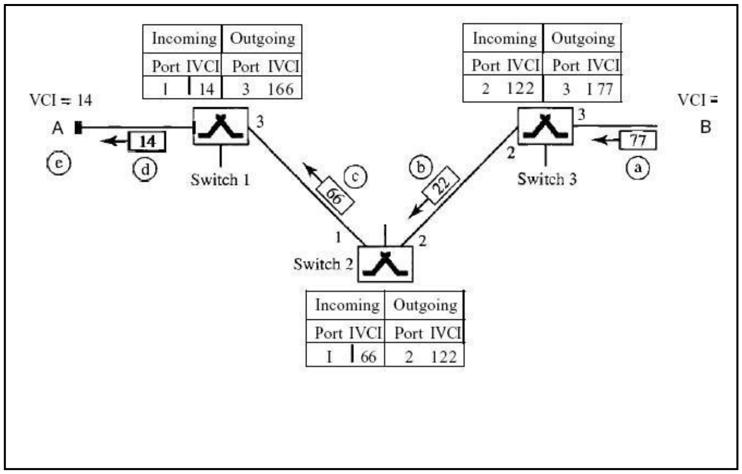


Fig 49 Setup acknowledgments in a virtual-circuit network

Teardown Phase

In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request.* Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

Efficiency

As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays. There is one big advantage in a virtual-circuit network even if resource allocation is on demand. The source can check the availability of the resources, without actually reserving it. Consider a family that wants to dine at a restaurant. Although the restaurant may not accept reservations (allocation of the tables is on demand), the family can call and find out the waiting time. This can save the family time and effort.

Delay in Virtual-Circuit Networks

In a virtual-circuit network, there is a one-time delay for setup and a one-time delay for teardown. If resources are allocated during the setup phase, there is no wait time for individual packets. Below Figure shows the delay for a packet traveling through two switches in a virtual- circuit network.

12. Packet Switching

Packet switching is a method of transferring the data to a network in form of packets. In order to transfer the file fast and efficient manner over the network and minimize the transmission latency, the data is broken into small pieces of variable length, called Packet. At the destination, all these small-parts (packets) has to be reassembled, belonging to the same file. A packet composes of payload and various control information. No pre-setup or reservation of resources is needed.

Packet Switching uses Store and Forward technique while switching the packets; while forwarding the packet each hop first store that packet then forward. This technique is very beneficial because packets may get discarded at any hop due to some reason. More than one path is possible between a pair of source and destination. Each packet contains Source and destination address using which they independently travel through the network. In other words, packets belonging to the same file may or may not travel through the same path. If there is congestion at some path, packets are allowed to choose different path possible over existing network.

Packet-Switched networks were designed to overcome the weaknesses of Circuit-Switched networks since circuit-switched networks were not very effective for small messages.

Advantage of Packet Switching over Circuit Switching :

- More efficient in terms of bandwidth, since the concept of reserving circuit is not there.
- Minimal transmission latency.
- More reliable as destination can detect the missing packet.
- More fault tolerant because packets may follow different path in case any link is down, Unlike Circuit Switching.
- Cost effective and comparatively cheaper to implement.

Disadvantage of Packet Switching over Circuit Switching :

- Packet Switching don't give packets in order, whereas Circuit Switching provides ordered delivery of packets because all the packets follow the same path.
- Since the packets are unordered, we need to provide sequence numbers to each packet.
- Complexity is more at each node because of the facility to follow multiple path.
- Transmission delay is more because of rerouting.
- Packet Switching is beneficial only for small messages, but for bursty data (large messages) Circuit Switching is better.

Modes of Packet Switching :

1. Connection-oriented Packet Switching (Virtual Circuit) :- Before starting the transmission, it establishes a logical path or virtual connection using signal ling protocol, between sender and receiver and all packets belongs to this flow will follow this predefined route. Virtual Circuit ID is provided by switches/routers to uniquely identify this virtual connection. Data is divided into small units and all these small units are appended with help of sequence number. Overall, three phases takes place here- Setup, data transfer and tear down phase. All address information is only transferred during setup phase. Once the route to destination is discovered, entry is added to switching table of each intermediate node. During data transfer, packet header (local header) may contain information such as length, timestamp, sequence number etc.

   Connection-oriented switching is very useful in switched WAN. Some popular protocols which use Virtual Circuit Switching approach are X.25, Frame-Relay, ATM and MPLS(Multi-Protocol Label Switching).

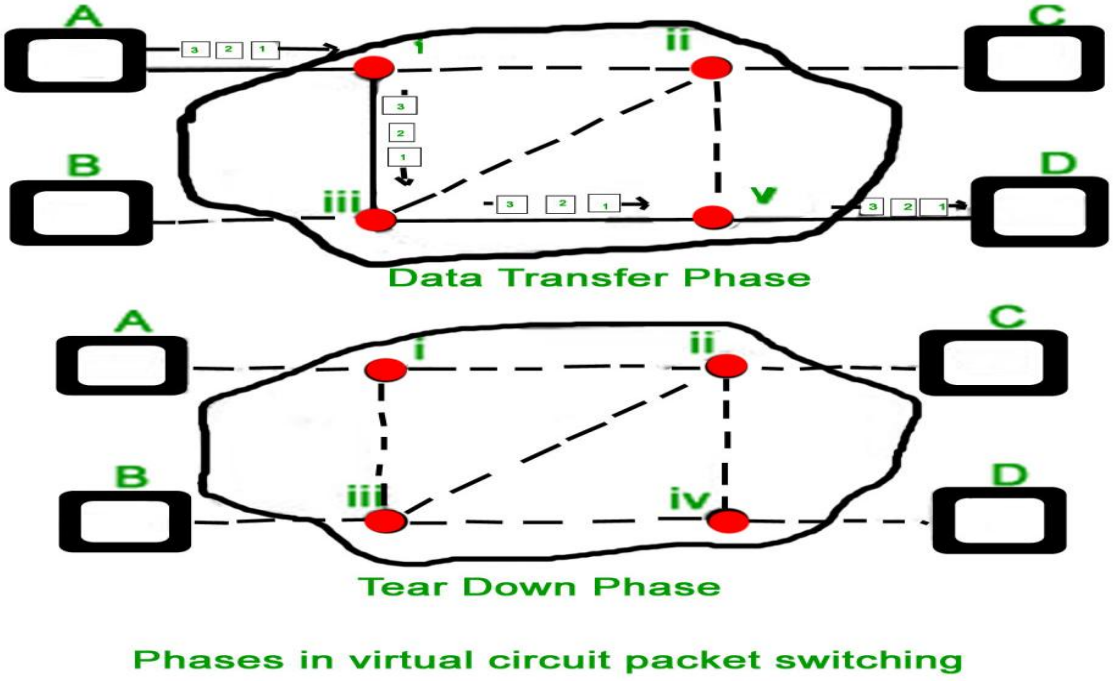


Fig 50 Packet Switching

2. Connectionless Packet Switching (Datagram) :- Unlike Connection-oriented packet switching, In Connectionless Packet Switching each packet contains all necessary addressing information such as source address, destination address and port numbers etc. In Datagram Packet Switching, each packet is treated independently. Packets belonging to one flow may take different routes because routing decisions are made dynamically, so the packets arrived at destination might be out of order. It has no connection setup and teardown phase, like Virtual Circuits.

3. Delays in Packet switching :

1. Transmission Delay
2. Propagation Delay
3. Queuing Delay
4. Processing Delay

Transmission Delay :

Time taken to put a packet onto link. In other words, it is simply time required to put data bits on the wire/communication medium. It depends on length of packet and bandwidth of network.

Transmission Delay = Data size / bandwidth = (L/B) second

Propagation delay :

Time taken by the first bit to travel from sender to receiver end of the link. In other words, it is simply the time required for bits to reach the destination from the start point. Factors on which Propagation delay depends are Distance and propagation speed.

Propagation delay = distance/transmission speed = d/s

Queuing Delay :

Queuing delay is the time a job waits in a queue until it can be executed. It depends on congestion. It is the time difference between when the packet arrived Destination and when the packet data was processed or executed. It may be caused by mainly three reasons i.e. originating switches, intermediate switches or call receiver servicing switches.

Average Queuing delay = (N-1)L/(2*R)

where N = no. of packets

L=size of packet

R=bandwidth

Processing Delay :

Processing delay is the time it takes routers to process the packet header. Processing of packets helps in detecting bit-level errors that occur during transmission of a packet to the destination. Processing delays in high-speed routers are typically on the order of microseconds or less.

13. Message Switching

Message switching was a technique developed as an alternate to circuit switching, before packet switching was introduced. In message switching, end users communicate by sending and receiving *messages* that included the entire data to be shared. Messages are the smallest individual unit.

Also, the sender and receiver are not directly connected. There are a number of intermediate nodes transfer data and ensure that the message reaches its destination. Message switched data networks are hence called hop-by-hop systems.

They provide 2 distinct and important characteristics:

1. **Store and forward –** The intermediate nodes have the responsibility of transferring the entire message to the next node. Hence, each node must have storage capacity. A message will only be delivered if the next hop and the link connecting it are both available, otherwise it'll be stored indefinitely. A store-and-forward switch forwards a message only if sufficient resources are available and the next hop is accepting data. This is called the store-and-forward property.
2. **Message delivery –** This implies wrapping the entire information in a single message and transferring it from the source to the destination node. Each message must have a header that contains the message routing information, including the source and destination.

Message switching network consists of transmission links (channels), store-and-forward switch nodes and end stations as shown in the following picture:

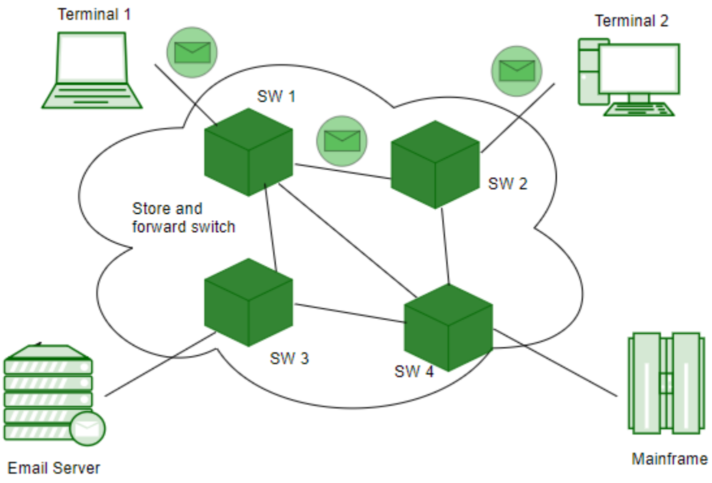


Fig 51 Message Switching

Characteristics of message switching

Message switching is advantageous as it enables efficient usage of network resources. Also, because of the store-and-forward capability of intermediary nodes, traffic can be efficiently regulated and controlled. Message delivery as one unit, rather than in pieces, is another benefit.

However, message switching has certain disadvantages as well. Since messages are stored indefinitely at each intermediate node, switches require large storage capacity. Also, these are pretty slow. This is because at each node, first there us wait till the entire message is received, then it must be stored and transmitted after processing the next node and links to it depending on availability and channel traffic. Hence, message switching cannot be used for real time or interactive applications like video conference.

Advantages of Message Switching

Message switching has the following advantages:

1. As message switching is able to store the message for which communication channel is not available, it helps in reducing the traffic congestion in network.

2. In message switching, the data channels are shared by the network devices.
3. It makes the traffic management efficient by assigning priorities to the messages.

Disadvantages of Message Switching

Message switching has the following disadvantages:

1. Message switching cannot be used for real time applications as storing of messages causes delay.
2. In message switching, message has to be stored for which every intermediate devices in the network requires a large storing capacity.

Applications

The store-and-forward method was implemented in telegraph message switching centres. Today, although many major networks and systems are packet-switched or circuit switched networks, their delivery processes can be based on message switching. For example, in most electronic mail systems the delivery process is based on message switching, while the network is in fact either circuit-switched or packet-switched.

14. Connection Oriented Services

There is a sequence of operation to be followed by the users of connection oriented service. These are:

1. Connection is established.
2. Information is sent.
3. Connection is released.

In connection oriented service we have to establish a connection before starting the communication. When connection is established, we send the message or the information and then we release the connection.

Connection oriented service is more reliable than connectionless service. We can send the message in connection oriented service if there is an error at the receivers end. Example of connection oriented is TCP (Transmission Control Protocol) protocol.

Connection Less Services

It is similar to the postal services, as it carries the full address where the message (letter) is to be carried. Each message is routed independently from source to destination. The order of message sent can be different from the order received.

In connectionless the data is transferred in one direction from source to destination without checking that destination is still there or not or if it prepared to accept the message. Authentication is not needed in this. Example of Connectionless service is UDP (User Datagram Protocol) protocol.

Difference: Connection oriented and Connectionless service

1. In connection oriented service authentication is needed, while connectionless service does not need any authentication.
2. Connection oriented protocol makes a connection and checks whether message is received or not and sends again if an error occurs, while connectionless service protocol does not guarantees a message delivery.
3. Connection oriented service is more reliable than connectionless service.
4. Connection oriented service interface is stream based and connectionless is message based.

Service Primitives

A service is formally specified by a set of primitives (operations) available to a user process to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets. The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connection-less service. There are five types of service primitives :

1. **LISTEN :** When a server is ready to accept an incoming connection it executes the LISTEN primitive. It blocks waiting for an incoming connection.
2. **CONNECT :** It connects the server by establishing a connection. Response is awaited.
3. **RECIEVE:** Then the RECIEVE call blocks the server.
4. **SEND :** Then the client executes SEND primitive to transmit its request followed by the execution of RECIEVE to get the reply. Send the message.
5. **DISCONNECT :** This primitive is used for terminating the connection. After this primitive one can't send any message. When the client sends DISCONNECT packet then the server also sends the DISCONNECT packet to acknowledge the client. When the server package is received by client then the process is terminated.

Connection Oriented Service Primitives

There are 5 types of primitives for Connection Oriented Service :

LISTEN	Block waiting for an incoming connection
CONNECTION	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Sending a message to the peer
DISCONNECT	Terminate a connection

Connectionless Service Primitives

There are 4 types of primitives for Connectionless Oriented Service:

UNIDATA	This primitive sends a packet of data
FACILITY, REPORT	Primitive for enquiring about the performance of the network, like delivery statistics.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT-IV- SOFTWARE TESTING – SBS1604

UNIT 4

Acceptance and operational testing: Defining the acceptance criteria - Developing an Acceptance plan - Executing the acceptance the acceptance plan - Developing Test Cases - Developing and updating test plan and test data.

Acceptance and Operational Testing

- ❑ Acceptance testing is formal testing conducted to determine whether a software system satisfies its acceptance criteria and to enable the buyer to determine whether to accept the system.
- ❑ Software acceptance testing at delivery is usually the final opportunity for the buyer to examine the software and to seek redress from the developer for insufficient or incorrect software.
- ❑ The customer/users of the soft-ware have one of three decisions to make:
- ❖ 1. The software system is usable as is and can be placed into a production state.
- ❖ 2. The software has some deficiencies; when corrected, the software can be placed into an operational state.
- ❖ 3. The software is severely deficient and it may or may not ever be placed into an operational state depending on the type of deficiencies, the alternatives available to the customer/users, and the cost to correct the deficiencies.

Acceptance Testing

- ❑ Software acceptance is an incremental process of approving or rejecting software systems during development or maintenance, according to how well the software satisfies predefined criteria
- ❑ Formal final software acceptance testing must occur at the end of the development process. It consists of tests to determine

whether the developed system meets predetermined functionality, performance, quality, and interface criteria.

Defining the Acceptance Criteria

- ❑ The user must assign the criteria the software must meet to be deemed acceptable.
- ❑ In preparation for developing the acceptance criteria, the user should do the following:
- ❖ Acquire full knowledge of the application for which the system is intended.
- ❖ Become fully acquainted with the application as it is currently implemented by the user's organization.
- ❖ Understand the risks and benefits of the development methodology that is to be used in correcting the software system.
- ❖ Fully understand the consequences of adding new functions to enhance the system.
- ❑ Acceptance requirements that a system must meet can be divided into these four categories:
- ❖ **Functionality. Internal consistency of documents and code and between stages;** traceability of functionality; adequate verification of logic; functional evaluation and testing; preservation of functionality in the operating environment.
- ❖ **Performance. Feasibility analysis of performance requirements; correct** simulation and instrumentation tools; performance analysis in the operating environment.

- ❖ **Interface quality. Interface documentation; interface complexity; interface and** integration test plans; interface ergonomics; operational environment interface testing.
- ❖ **Overall software quality. Quantification of quality measures; criteria for** acceptance of all software products; adequacy of documentation and software system development standards; quality criteria for operational testing.

- ❑ Assessing the criticality of a system is important in determining quantitative acceptance criteria. By definition, all safety criteria are critical; and by law, certain security requirements are critical.
- ❑ Some typical factors affecting criticality include the following:

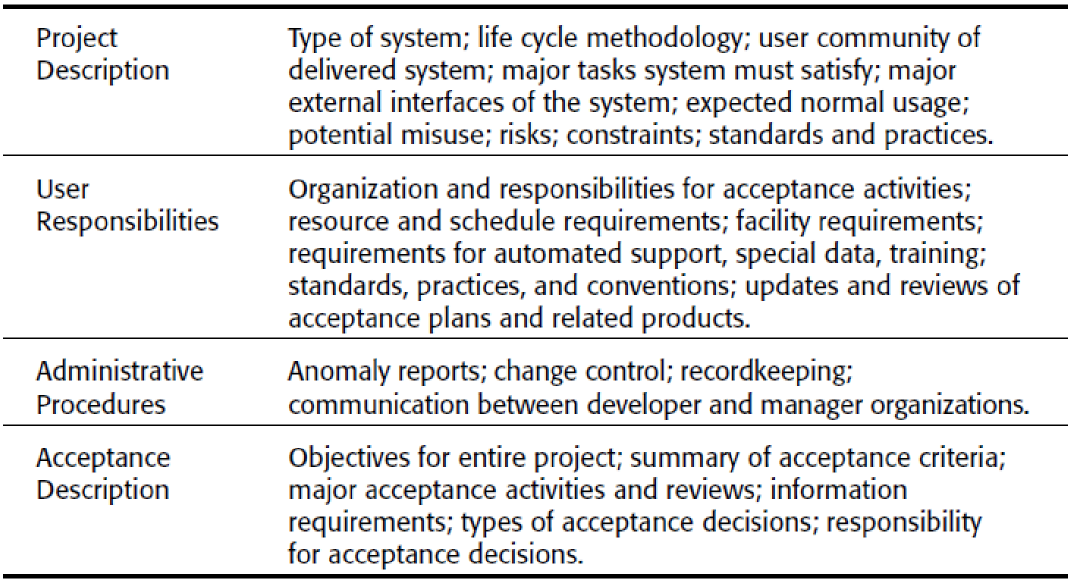
- ❖ Importance of the system to organization or industry
- ❖ Consequence of failure
- ❖ Complexity of the project
- ❖ Technology risk
- ❖ Complexity of the user environment

Developing an Acceptance Plan

- ❑ The first step to achieving software acceptance is the simultaneous development of a software acceptance plan, general project plans, and software requirements to ensure that user needs are represented correctly and completely.
- ❑ This simultaneous development will provide an overview of the acceptance activities, to ensure that resources for them are included in the project plans.

- Note that the initial plan may not be complete and may contain estimates that will need to be changed as more complete project information becomes available.
- After the initial software acceptance plan has been prepared, reviewed, and approved, the acceptance manager is responsible for implementing the plan and for ensuring that the plan's objectives are met.
- It may have to be revised before this assurance is warranted.

Acceptance Plan Contents



Project Description	Type of system; life cycle methodology; user community of delivered system; major tasks system must satisfy; major external interfaces of the system; expected normal usage; potential misuse; risks; constraints; standards and practices.
User Responsibilities	Organization and responsibilities for acceptance activities; resource and schedule requirements; facility requirements; requirements for automated support, special data, training; standards, practices, and conventions; updates and reviews of acceptance plans and related products.
Administrative Procedures	Anomaly reports; change control; recordkeeping; communication between developer and manager organizations.
Acceptance Description	Objectives for entire project; summary of acceptance criteria; major acceptance activities and reviews; information requirements; types of acceptance decisions; responsibility for acceptance decisions.

- The plan must include the techniques and tools that will be utilized in acceptance testing.
- Normally, testers will use the organization's current testing tools, which should be oriented toward specific testing techniques.
- Two categories of testing techniques can be used in acceptance testing: structural and functional.

❑ The functional testing techniques help ensure that the requirements/specifications are properly satisfied by the software system.

❑ Functional testing is not concerned with how processing occurs, but with the results of processes.

❑ Structural testing ensures sufficient checking of the implementation of the function by finding test data that will force sufficient coverage of the structured presence in the implemented software. It evaluates all aspects of this structure to verify that the structure is sound.

Executing the Acceptance Plan

❑ The objective of this step is to determine whether the acceptance criteria have been met in a delivered product.

❑ This can be accomplished through reviews, which involve looking at interim products and partially developed deliverables at various points through-out the developmental process.

❑ It can also involve testing the executable software system.

❑ The determination of which (or both) of these techniques to use will depend on the criticality of the software, the size of the software program, the resources involved, and the time period over which the software is being developed.

❑ Software acceptance criteria should be specified in the formal project plan.

❑ The plan identifies products to be tested, the specific pass/fail criteria, the reviews, and the types of testing that will occur throughout the entire life cycle.

❑ Acceptance decisions need a framework in which to operate; items such as contracts, acceptance criteria, and formal mechanisms are part of this framework.

❑ Software acceptance must state or refer to specific criteria that products must meet to be accepted.

❑ The principal means of reaching acceptance in the development of critical software systems is to hold a periodic review of interim software documentation and other software products.

❑ A disciplined acceptance program for software of any type may include reviews as a formal mechanism.

❑ When the acceptance decision requires change, another review becomes necessary to ensure that the required changes have been properly configured and implemented and that any affected segments are acceptable.

❑ For large or complicated projects, several reviews may be necessary during the development of a single product.

❑ Some software acceptance activities may include testing pieces of the software; for-mal software acceptance testing occurs at the point in the development life cycle when the user accepts or rejects the software.

❑ This means a contractual requirement between the user and the project team has been met.

❑ Rejection normally means additional work must be done on the system to render it acceptable to the user.

❑ Final software acceptance testing is the last opportunity for the user to examine the software for functional, interface, performance, and quality features prior to the final acceptance review.

- The system at this time must include the delivered software, all user documentation, and final versions of other software deliverables.

Developing Test Cases (Use Cases) Based on How Software Will Be Used

- Incomplete, incorrect, and missing test cases can cause incomplete and erroneous test results, which, at minimum, means that rework is necessary, and at worst, means that a flawed system is developed.
- It is necessary to ensure that all required test cases are identified so that all system functionality requirements are tested.
- A use case is a description of how a user (or another system) uses the system being designed to perform a given task.
- A system is described by the sum of its use cases.
- Each instance or scenario of a use case will correspond to one test case.
- Incorporating the use case technique into the development life cycle will address the effects of incomplete, incorrect, and missing test cases.
- Use cases represent an easy-to-use approach applicable to both conventional and object-oriented system developments.
- Use cases provide a powerful means of communication between customers, developers, testers, and other project personnel.
- Test cases can be developed with system users and designers as the use cases are being developed. Having the test cases this early in the project provides a baseline for the early planning of acceptance testing.

❑ Another advantage of having test cases early on is that if a packaged software solution is indicated, the customer can use them to evaluate purchased software earlier in the development cycle.

❑ Using the use case approach will ensure not only meeting requirements but also expectations.

❑ **Building a System Boundary Diagram**

❖ A system boundary diagram depicts the interfaces between the software being tested and the individuals, systems, and other interfaces.

❖ These interfaces or external agents in this work practice will be referred to as “actors.”

❖ The purpose of the system boundary diagram is to establish the scope of the system and to identify the actors (i.e., the interfaces) that need to be developed.

❖ An example of a system boundary diagram for an automatic automated teller machine for an organization called “Best Bank” is illustrated in Figure. 4.1

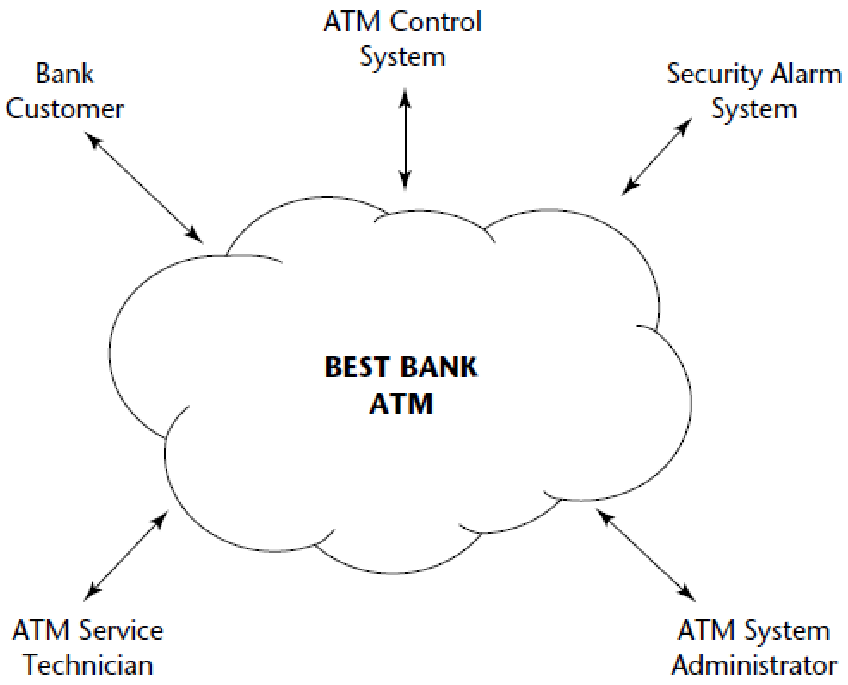
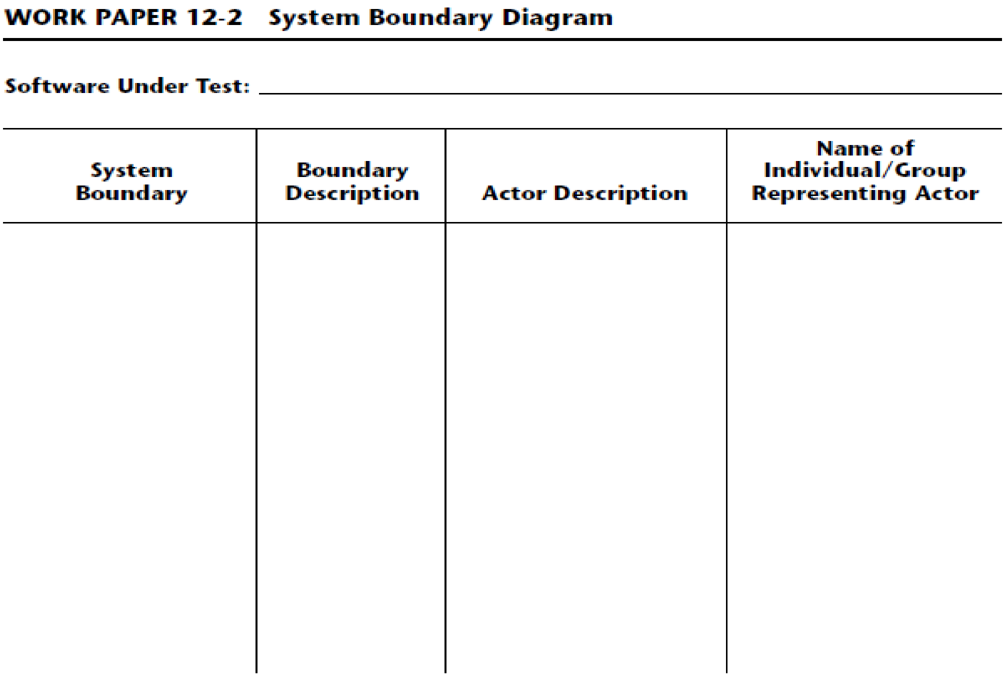


Figure. 4. 1: system boundary diagram for an automatic automated teller machine

- Work Paper 12-2 is designed to document a system boundary diagram for the software under test. For that software, each system boundary needs to be defined.



WORK PAPER 12-2 System Boundary Diagram

Software Under Test: ____________________

System Boundary	Boundary Description	Actor Description	Name of Individual/Group Representing Actor

Figure. 4.2: work paper 12-2

❑ System boundaries can include the following:

❖ Individuals/groups that manually interface with the software.

❖ Other systems that interface with the software.

❖ Libraries.

❖ Objects within object-oriented systems.

❖ Each system boundary should be described. For each boundary, an actor must be identified.

❑ Two aspects of actor definition are required. The first is the actor description, and the second is the name of an individual or group who can play the role of the actor (i.e., rep-resent that boundary interface).

❑ For example, in Figure, the security alarm system is identified as an interface.

❑ The actor is the security alarm company.

❑ The name of a person in the security alarm company or the name of someone who can represent the security alarm company must be identified.

❑ Note that in some instances the actor and the individual may be the same, such as the ATM system administrator listed in Figure

❑ **Defining Use Cases**

❑ An individual use case consists of the following:

❖ Preconditions that set the stage for the series of events that should occur for the use case

❖ Post-conditions that state the expected outcomes of the preceding process

❖ Sequential narrative of the execution of the use case

❑ Use cases are used to do the following:

❖ Manage (and trace) requirements

❖ Identify classes and objects (OO)

❖ Design and code (non-OO)

❖ Develop application documentation

❖ Develop training

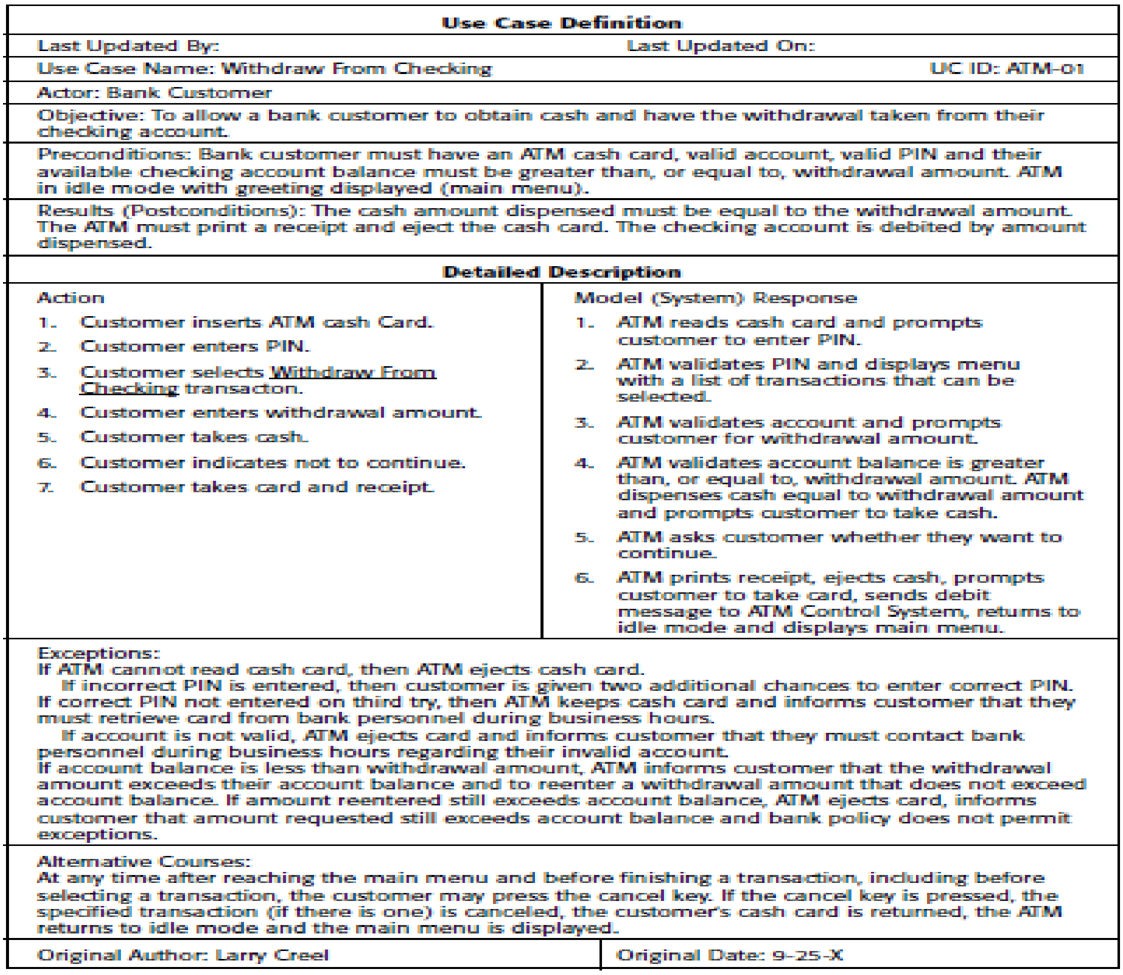
❖ Develop test cases

❑ The use case definition is done by the actor. The actor represents the system boundary interface and prepares all the use cases for that system boundary interface.

❑ Note that this can be done by a single individual or a team of individuals.

❑ Work Paper 12-3 is used for defining a use case. An example of a completed Work Paper 12-3 for an ATM system is illustrated in Figure

❑ This example is for an ATM system. The case is a bank customer making a withdrawal from their checking account on an ATM.



Use Case Definition

Last Updated By: Last Updated On:

Use Case Name: Withdraw From Checking UC ID: ATM-01

Actor: Bank Customer

Objective: To allow a bank customer to obtain cash and have the withdrawal taken from their checking account.

Preconditions: Bank customer must have an ATM cash card, valid account, valid PIN and their available checking account balance must be greater than, or equal to, withdrawal amount. ATM in idle mode with greeting displayed (main menu).

Results (Postconditions): The cash amount dispensed must be equal to the withdrawal amount. The ATM must print a receipt and eject the cash card. The checking account is debited by amount dispensed.

Detailed Description

Action	Model (System) Response
1. Customer inserts ATM cash Card. 2. Customer enters PIN. 3. Customer selects <u>Withdraw From Checking</u> transacton. 4. Customer enters withdrawal amount. 5. Customer takes cash. 6. Customer indicates not to continue. 7. Customer takes card and receipt.	1. ATM reads cash card and prompts customer to enter PIN. 2. ATM validates PIN and displays menu with a list of transactions that can be selected. 3. ATM validates account and prompts customer for withdrawal amount. 4. ATM validates account balance is greater than, or equal to, withdrawal amount. ATM dispenses cash equal to withdrawal amount and prompts customer to take cash. 5. ATM asks customer whether they want to continue. 6. ATM prints receipt, ejects cash, prompts customer to take card, sends debit message to ATM Control System, returns to idle mode and displays main menu.

Exceptions:
If ATM cannot read cash card, then ATM ejects cash card.

If incorrect PIN is entered, then customer is given two additional chances to enter correct PIN. If correct PIN not entered on third try, then ATM keeps cash card and informs customer that they must retrieve card from bank personnel during business hours.

If account is not valid, ATM ejects card and informs customer that they must contact bank personnel during business hours regarding their invalid account.
If account balance is less than withdrawal amount, ATM informs customer that the withdrawal amount exceeds their account balance and to reenter a withdrawal amount that does not exceed account balance. If amount reentered still exceeds account balance, ATM ejects card, informs customer that amount requested still exceeds account balance and bank policy does not permit exceptions.

Alternative Courses:
At any time after reaching the main menu and before finishing a transaction, including before selecting a transaction, the customer may press the cancel key. If the cancel key is pressed, the specified transaction (if there is one) is canceled, the customer's cash card is returned, the ATM returns to idle mode and the main menu is displayed.

Original Author: Larry Creel Original Date: 9-25-X

Figure .4.3: wok paper 12-3 for an atm system

❑ Developing Test Cases

❑ A test case is a set of test inputs, execution conditions, and expected results developed for a particular test objective.

❑ There should be a one-to-one relationship between use case definitions and test cases.

❑ There needs to be at least two test cases for each use case: one for successful execution of the use case and one for an unsuccessful execution of a test case. However, there may be numerous test cases for each use case.

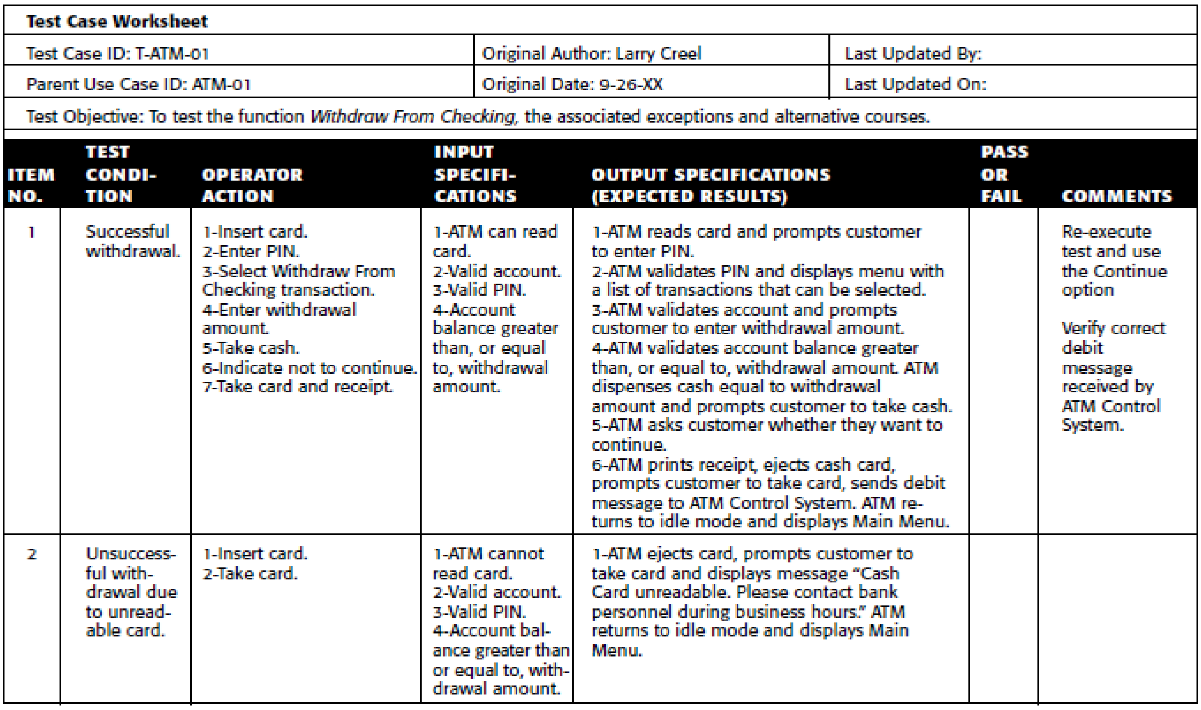
❑ Additional test cases are derived from the exceptions and alternative courses of the use case. Note that additional detail may need to be added to support the actual test-ing of all the possible scenarios of the use case.

❑ The use case description is the input into Work Paper 12-4. The actor who prepared the use case description also prepares the test case work paper.

❑ There will be at least two test conditions for each use case description and normally many more.

❑ The actor tries to determine all of the possible scenarios that occur for each use case. Figure is an example of a test case work paper designed to test the function “withdrawal from checking from an ATM.” Note that this is Action 3 from Figure 4.3. Work Paper 12-1 becomes the input for Work Paper 12-4, as shown in Figure 4.5.

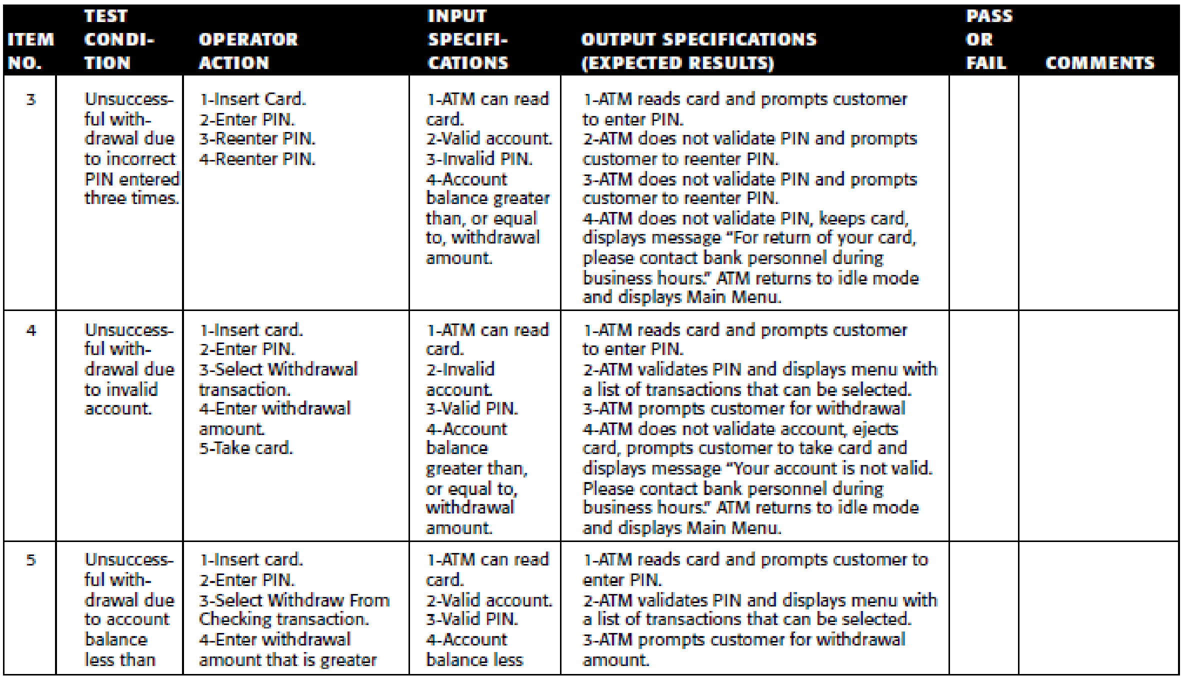
❑ After acceptance testing, a decision must be made for each acceptance criterion as to whether it has been achieved.



Test Case Worksheet		
Test Case ID: T-ATM-01	Original Author: Larry Creel	Last Updated By:
Parent Use Case ID: ATM-01	Original Date: 9-26-XX	Last Updated On:
Test Objective: To test the function *Withdraw From Checking*, the associated exceptions and alternative courses.		

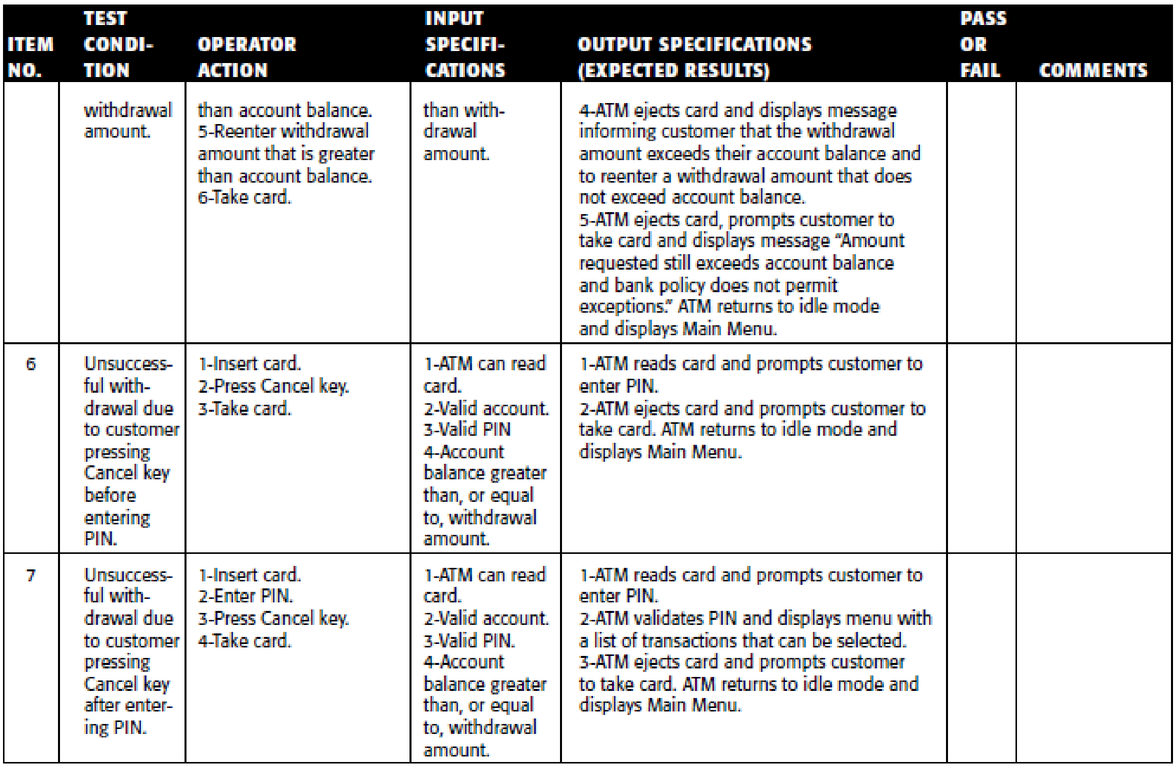
ITEM NO.	TEST CONDITION	OPERATOR ACTION	INPUT SPECIFICATIONS	OUTPUT SPECIFICATIONS (EXPECTED RESULTS)	PASS OR FAIL	COMMENTS
1	Successful withdrawal.	1-Insert card. 2-Enter PIN. 3-Select Withdraw From Checking transaction. 4-Enter withdrawal amount. 5-Take cash. 6-Indicate not to continue. 7-Take card and receipt.	1-ATM can read card. 2-Valid account. 3-Valid PIN. 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM validates PIN and displays menu with a list of transactions that can be selected. 3-ATM validates account and prompts customer to enter withdrawal amount. 4-ATM validates account balance greater than, or equal to, withdrawal amount. ATM dispenses cash equal to withdrawal amount and prompts customer to take cash. 5-ATM asks customer whether they want to continue. 6-ATM prints receipt, ejects cash card, prompts customer to take card, sends debit message to ATM Control System. ATM returns to idle mode and displays Main Menu.		Re-execute test and use the Continue option Verify correct debit message received by ATM Control System.
2	Unsuccessful withdrawal due to unreadable card.	1-Insert card. 2-Take card.	1-ATM cannot read card. 2-Valid account. 3-Valid PIN. 4-Account balance greater than or equal to, withdrawal amount.	1-ATM ejects card, prompts customer to take card and displays message "Cash Card unreadable. Please contact bank personnel during business hours." ATM returns to idle mode and displays Main Menu.		

Figure .4.4: Example of Completed work paper 12-4 for an ATM withdrawal



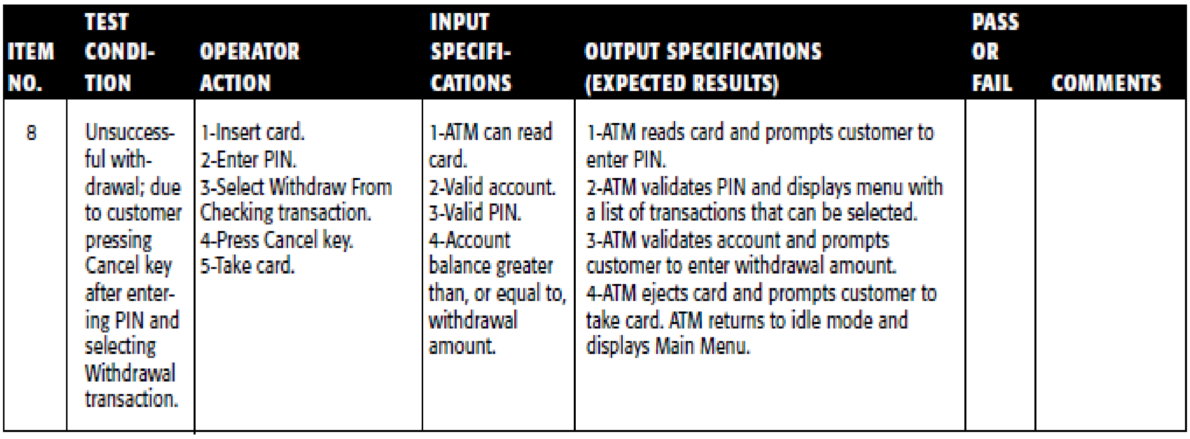
ITEM NO.	TEST CONDITION	OPERATOR ACTION	INPUT SPECIFICATIONS	OUTPUT SPECIFICATIONS (EXPECTED RESULTS)	PASS OR FAIL	COMMENTS
3	Unsuccessful withdrawal due to incorrect PIN entered three times.	1-Insert Card. 2-Enter PIN. 3-Reenter PIN. 4-Reenter PIN.	1-ATM can read card. 2-Valid account. 3-Invalid PIN. 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM does not validate PIN and prompts customer to reenter PIN. 3-ATM does not validate PIN and prompts customer to reenter PIN. 4-ATM does not validate PIN, keeps card, displays message "For return of your card, please contact bank personnel during business hours." ATM returns to idle mode and displays Main Menu.		
4	Unsuccessful withdrawal due to invalid account.	1-Insert card. 2-Enter PIN. 3-Select Withdrawal transaction. 4-Enter withdrawal amount. 5-Take card.	1-ATM can read card. 2-Invalid account. 3-Valid PIN. 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM validates PIN and displays menu with a list of transactions that can be selected. 3-ATM prompts customer for withdrawal 4-ATM does not validate account, ejects card, prompts customer to take card and displays message "Your account is not valid. Please contact bank personnel during business hours." ATM returns to idle mode and displays Main Menu.		
5	Unsuccessful withdrawal due to account balance less than	1-Insert card. 2-Enter PIN. 3-Select Withdraw From Checking transaction. 4-Enter withdrawal amount that is greater	1-ATM can read card. 2-Valid account. 3-Valid PIN. 4-Account balance less	1-ATM reads card and prompts customer to enter PIN. 2-ATM validates PIN and displays menu with a list of transactions that can be selected. 3-ATM prompts customer for withdrawal amount.		

Figure .4.4: Example of Completed work paper 12-4 for an ATM withdrawal



ITEM NO.	TEST CONDI-TION	OPERATOR ACTION	INPUT SPECIFI-CATIONS	OUTPUT SPECIFICATIONS (EXPECTED RESULTS)	PASS OR FAIL	COMMENTS
	withdrawal amount.	than account balance. 5-Reenter withdrawal amount that is greater than account balance. 6-Take card.	than with-drawal amount.	4-ATM ejects card and displays message informing customer that the withdrawal amount exceeds their account balance and to reenter a withdrawal amount that does not exceed account balance. 5-ATM ejects card, prompts customer to take card and displays message "Amount requested still exceeds account balance and bank policy does not permit exceptions." ATM returns to idle mode and displays Main Menu.		
6	Unsuccess-ful with-drawal due to customer pressing Cancel key before entering PIN.	1-Insert card. 2-Press Cancel key. 3-Take card.	1-ATM can read card. 2-Valid account. 3-Valid PIN 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM ejects card and prompts customer to take card. ATM returns to idle mode and displays Main Menu.		
7	Unsuccess-ful with-drawal due to customer pressing Cancel key after enter-ing PIN.	1-Insert card. 2-Enter PIN. 3-Press Cancel key. 4-Take card.	1-ATM can read card. 2-Valid account. 3-Valid PIN. 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM validates PIN and displays menu with a list of transactions that can be selected. 3-ATM ejects card and prompts customer to take card. ATM returns to idle mode and displays Main Menu.		

Figure .4.4: Example of Completed work paper 12-4 for an ATM withdrawal



ITEM NO.	TEST CONDI-TION	OPERATOR ACTION	INPUT SPECIFI-CATIONS	OUTPUT SPECIFICATIONS (EXPECTED RESULTS)	PASS OR FAIL	COMMENTS
8	Unsuccess-ful with-drawal; due to customer pressing Cancel key after enter-ing PIN and selecting Withdrawal transaction.	1-Insert card. 2-Enter PIN. 3-Select Withdraw From Checking transaction. 4-Press Cancel key. 5-Take card.	1-ATM can read card. 2-Valid account. 3-Valid PIN. 4-Account balance greater than, or equal to, withdrawal amount.	1-ATM reads card and prompts customer to enter PIN. 2-ATM validates PIN and displays menu with a list of transactions that can be selected. 3-ATM validates account and prompts customer to enter withdrawal amount. 4-ATM ejects card and prompts customer to take card. ATM returns to idle mode and displays Main Menu.		

Figure 4.5: Acceptance Criteria

- ❑ **Reaching an Acceptance Decision**
- ❑ Final acceptance of software based on acceptance testing usually means that the software project has been completed, except for any caveats or contingencies.

❑ Final acceptance for the software occurs, and the developer has no further development obligations (except, of course, for maintenance, which is a separate issue).

❑ Typical acceptance decisions include the following:

❖ Required changes are accepted before progressing to the next activity.

❖ Some changes must be made and accepted before further development of that section of the product; other changes may be made and accepted at the next major review.

❖ Progress may continue and changes may be accepted at the next review.

❖ No changes are required and progress may continue.

❑ The goal is to achieve and accept "perfect" software, but usually, some criteria will not be completely satisfied with each product, in which case the user may choose to accept less-than-perfect software.

Developing and Updating the Test Plan

❑ The test plan for software maintenance is a shorter, more directed version of a test plan used for a new application system. Whereas new application testing can take many weeks or months, software maintenance testing often must be done within a single day or a few hours.

❑ Because of time constraints, many of the steps that might be performed individually in a new system are combined or condensed into a short time span.

❑ This increases the need for planning so that all aspects of the test can be executed within the allotted time.

- ❑ The types of testing will vary based upon the implemented change. For example, if a report is modified, there is little need to test recovery and backup plans. On the other hand, if new files are created or processing procedures changed, restart and recovery should be tested.

- ❑ The preparation of a test plan is a two-part process. The first part is the determination of what types of tests should be conducted, and the second part is the plan for how to conduct them. Both parts are important in software maintenance testing.

- ❖ Elements to be tested (types of testing) are as follows:
- ❖ Changed transactions
- ❖ Changed programs
- ❖ Operating procedures
- ❖ Control group procedures
- ❖ User procedures
- ❖ Intersystem connections
- ❖ Job control language
- ❖ Interface to systems software
- ❖ Execution of interface to software systems
- ❖ Security
- ❖ Backup/recovery procedures

Developing and Updating the Test Data

- ❑ Data must be prepared for testing all the areas changed during a software maintenance process.

- For many applications, the existing test data will be sufficient to test the new change. However, in many situations, new test data will need to be prepared.
- In some cases, the preparation of test data can be significantly different for software maintenance than for new systems.
- For example, when the system is operational it may be possible to test the application in a live operational mode, thus eliminating the need for technical test data, and enabling maintenance software analysts to use the same input the users of the application prepare.
- Special accounts can be established to accumulate test data processed during testing in a production mode.
- The information in these accounts can then be eliminated after the test, which negates the effect of entering test data into a production environment.
- It is important to test both what should be accomplished, as well as what can go wrong.
- Most tests do a good job of verifying that the specifications have been implemented properly. Where testing frequently is inadequate is in verifying the unanticipated conditions. Included in this category are the following:

- Transactions with erroneous data
- Unauthorized transactions

- ❖ Transactions entered too early
- ❖ Transactions entered too late
- ❖ Transactions that do not correspond with master data contained in the application
- ❖ Grossly erroneous transactions, such as transactions that do not belong to the
- ❖ application being tested
- ❖ Transactions with larger values in the fields than anticipated

- ❑ These types of transactions can be designed by doing a simple risk analysis scenario.
- ❑ The risk analysis scenario involves brainstorming with key people involved in the application, such as users, maintenance systems analysts, and auditors.
- ❑ These people attempt to ask all the questions, such as, “What if this type of error were entered? What would happen if too large a value were entered in this field?”
- ❑ The three methods that can be used to develop/update test data are as follows:

- ❖ **Update existing test data. If test files have been created for a previous version,** they can be used for testing a change. However, the test data will need to be updated to reflect the changes to the software. Note that testers may wish to use both versions in conducting testing. Version 1 is to test that the unchanged portions perform now as they did in the previous versions. The new version is to test the changes. Updating the test data should follow the same processes used in creating new test data.

- ❖ **Create new test data. The creation of new test data for maintenance follows the** same methods as creating test data for a new software system.

- ❖ **Use production data for testing. Tests are performed using some or all of the** production data for test purposes (date-modified, of course), particularly when there are no functional changes. Using production data for test purposes may result in the following impediments to effective testing:

- ➢ **Missing test transactions. The transaction types on a production data file** may be limited. For example, if the tester wants to test an override of a standard price, that transaction may not occur on the production data file.

- ➢ **Multiple tests of the same transaction. Production data usually represents** the production environment, in which 80 to 90 percent of the transactions are of approximately the same type. This means that some transaction types are not tested at all, while others are tested hundreds of times.

- ➢ **Unknown test results. An important part of testing is to validate that cor**rect results are produced. When testers create test transactions, they have control over the expected results. When production data is used, however, testers must manually calculate the correct processing results, perhaps causing them to misinterpret the intent of the transaction and thereby to misinterpret the results.

- **Lack of ownership. Production data is owned by the production area,** whereas test data created by testers is owned by the testers. Some testers are more involved and interested in test data they created themselves than in test data borrowed from another owner.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT-V - SOFTWARE TESTING – SBS1604

UNIT V

A **test strategy** is an outline that describes the testing approach of the software development cycle. It is created to inform project managers, testers, and developers about some key issues of the testing process. This includes the testing objective, methods of testing new functions, total time and resources required for the project, and the testing environment.

Test strategies describe how the product risks of the stakeholders are mitigated at the test-level, which types of testing are to be performed, and which entry and exit criteria apply. They are created based on development design documents. System design documents are primarily used, and occasionally conceptual design documents may be referred to. Design documents describe the functionality of the software to be enabled in the upcoming release. For every stage of development design, a corresponding test strategy should be created to test the new feature sets.

What is Test Strategy?

The answer lies within the question "How is a team is going to test the particular application?" This means that the exact process the team is going to follow in the testing environment should be mentioned. Most organizations follow a standard template but even without using one, it is possible to write a test strategy that is simple but effective.

But what's the difference between test plan and test strategy? Both terms can often confuse testers. A test plan is a vision as to what a team wants to achieve, whereas a test strategy is a roadmap designed to achieve that vision.

What should a Test Strategy Document include?

Scope and Overview

We will expand our discussion now to include testing activities and phases which need to be carried out, the overall project timeline as defined in the test plan, as well as information on who the document is for.

The Test Approach

In this section, the testing process, level of testing to be carried out on the product, and the roles and responsibilities of each team member are mentioned. It also elaborates every test type defined in the test plan (unit, integration, system, regression, smoke, usability, performance, etc.) as well as consideration of why the particular testing type should be employed, who will perform this testing activity, what will be the responsibilities of the tester, and details of any automation strategy and tools if applicable.

Test Environment

Here, information about the test environment setup needs to be clearly outlined, and the number of test environments required is also mentioned. For example, you might need a development environment where the application is tested before moving into the testing environment, or setting up a test environment for the functional test team and another environment for UAT team. This section contains all details of the team members allocated to each environment, software and hardware requirements for the environment(s), memory, OS, etc.

Tools for Testing

This section usually outlines the test management and automation tools required for test execution. In order to carry out performance, load, and security testing, the organization needs to list all tools and approaches needed.

Risk Analysis

Careful consideration of all possible risks will be outlined in this section. A detailed plan to alleviate those risks is provided to help reduce the possibility of failure at any point.

Review and Approval

When all the things have been described in detail, it’s time to review, then approve the test strategy document. A formal sign-off is required from all the main stakeholders of the project which will include project management, business analysis, the development and testing teams, and others according to the organization’s structure. A test strategy document is dynamic and can be changed at later stages whenever there arises a need to do so.

Rapid Application Development Testing

Rapid application development (RAD) is an effective software development paradigm that provides a systematic and automatable means of developing a software system under circumstances where initial requirements are not well known or where requirements change frequently during development.

Providing high software quality assurance requires sufficient software testing.

- ❑ This testing strategy assumes the RAD system
- ❖ is iterative.
- ❖ is evolutionary.
- ❖ contains RAD language with a defined grammar.
- ❖ provides reusable components capability (library and retrieval).
- ❖ uses implementation code from reusable components written in a high-level language.
- ❖ contains a sophisticated support environment.

Objective

- ❑ The RAD testing methodology described in this test process s designed to take maximum advantage of the iterative nature of RAD.
- ❑ It also should focus on the requirements-capturing purpose of developing.
- ❑ Thus, a RAD-based testing technique starts by capturing the testing information resident in the RAD process in a form suitable for thoroughly testing the RAD system.
- ❑ Testers must know both the assumptions and the requirements the designers are trying to meet so that a test series can be built to verify the system.
- ❑ Remember, the test personnel are not usually the design personnel, nor should they be; therefore, RAD-based testing must provide tools and methods to ana-lyze system requirements and capture requirement changes

Concerns

Testing Iterations

- ❑ The iterative nature of software development implies that the system must track revision histories and maintain version control of alternate RAD versions.
- ❑ The user’s response to a demonstration may require that the RAD fall back to a previous iteration for change, or the developer might wish to demonstrate several iteration alternatives for user comment (one of which, or portions of several being selected).
- ❑ Requirements may be added, changed, or deleted.

- ❑ Test goals must easily change to fit the modified requirements. Ideally, the developing environment will capture this modification explicitly, along with the accompanying purpose of the modification.
- ❑ Any testing tools developed must take these modifications into account to test the proper version and to
- ❑ appropriately consider the requirements and purpose germane to that version for test development.
- ❑ The tool should also exploit change as a likely source of errors. A tool that helps testers compare changes from one iteration to the next, along with system dependencies potentially affected by changes, helps test development considerably.

Testing Components

- ❑ The use of reusable components raises reliability concerns.
- ❑ Have the components been unit tested, and if so, to what degree?
- ❑ Have they been used in previous implementations, and if so, which ones?
- ❑ What testing has been conducted on the previous implementations as well as the individual components?
- ❑ The testing methodology must consider how information on past component testing can be recorded and referenced to determine what unit testing might still be needed and what integration testing strategies might best check the components in their instantiated context.

Testing Performance

- ❑ One necessary testing component is a set of test conditions. Requirements-based and functional testing base test conditions on some stated form of behavior or required performance standards, such as formal specifications or a requirements document.
- ❑ The development methodology does not provide a separate performance standard.
- ❑ The testing methodology must establish an objective standard of the intended behavior of the RAD under consideration. Every program must have an objective performance standard.
- ❑ The developing system must then, in some fashion, provide the tester and his or her tools with access to a system functionality description and system requirements to allow rapid, complete, and consistent derivation from the RAD.

- ❑ This access to functionality descriptions and requirements has the added benefit of helping develop scripts for demonstrations so that particular iteration changes and enhancements will be highlighted for the user's comments.

Recording Test Information

- ❑ The software development environment not only should capture requirements, assumptions, and design decisions but, ideally, should map these into the RAD in a way useful to both rapid application development and testing.
- ❑ This mapping automatically pro-vides a trace, documenting the RAD's development.
- ❑ As the system grows, knowing why a particular design decision was made and being able to see where (and how) the RAD implements it will be difficult without automated support.
- ❑ The developing/testing paradigm must capture mappings from design or development decisions to the implementation.
- ❑ These mappings need to be rapidly revisable to quickly make the next RAD iteration.

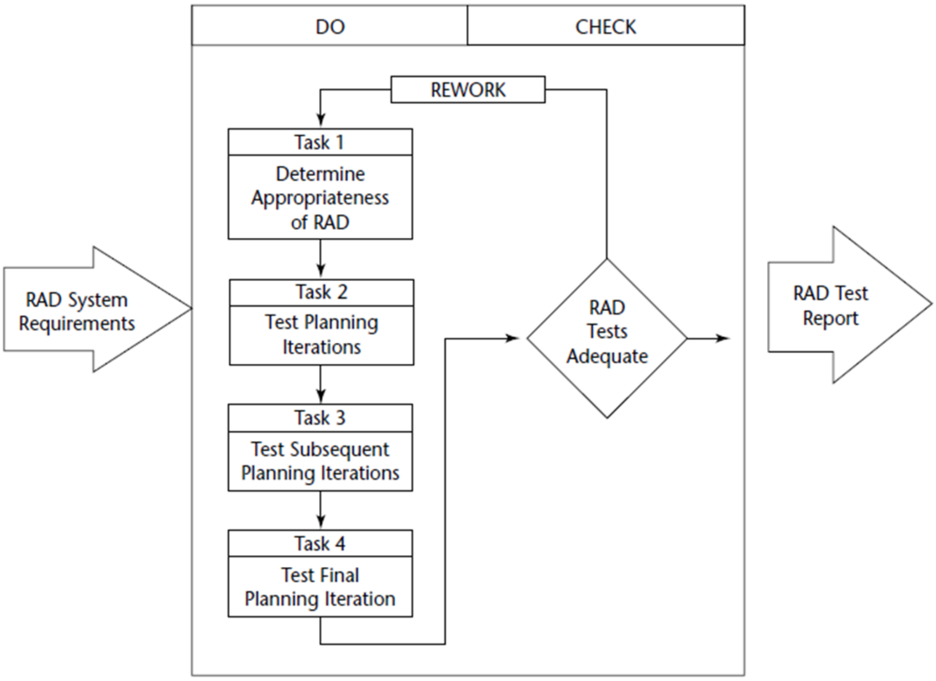
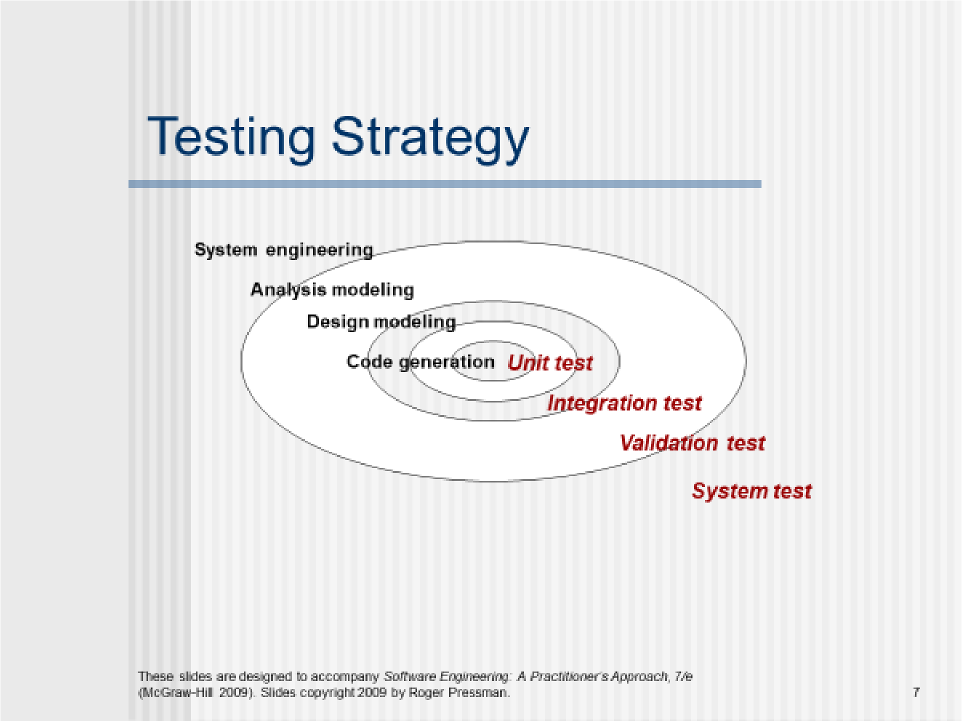


Fig. 5.1. Workbench for testing RAD systems

- ❑ The only input to this test process is the RAD requirements. Because of the nature of RAD, the requirements are normally incomplete when development begins.
- ❑ The requirements will be continually developed throughout various iterations.
- ❑ Thus, the input to each of the three steps in the recommended RAD test process will be different.

Testing Within Iterative RAD

- ❑ The most obvious approach to testing during RAD would be to treat each development iteration as one software life cycle.
- ❑ An advantage is that this keeps intact the methodology of testing familiar to most testers.
- ❑ The lack of a conventional specification effectively removes the information basis for test planning.
- ❑ Under current descriptions of the developing process, a specification would need to be generated, at least in part, before conventional techniques could be applied.
- ❑ The process's complexity is also compounded by the need to conduct a full cycle of testing for each iteration, even though the early iterations will almost never contain detailed or unchanging requirements. This would be inefficient and impractical testing.
- ❑ An alternative test approach is to iterate test planning in parallel with the develop-ing iterations.
- ❑ This will simplify testing and reduce overall testing costs when compared to the preceding approach.
- ❑ The initial test plan would consider only the basic system description contained in the initial RAD iteration.
- ❑ As RAD iterations proceed, the test plan would expand to incorporate the latest iteration modifications.
- ❑ One disadvantage is that this approach causes the test plan to follow the RAD process closely.
- ❑ The decisions in the RAD might unduly influence the test plan, causing important test conditions not to be explored.
- ❑ The possible disadvantage suggests that the unit and integration testing might be done iteratively, with acceptance testing occurring entirely on the final iteration of the development cycle.



Testing Strategy

System engineering

Analysis modeling

Design modeling

Code generation

Unit test

Integration test

Validation test

System test

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman. 7

Spiral Testing

Posted ON 6 Oct

Spiral testing is risk-based testing, where testers will take risk due many factors at the same time they will equipped with solutions for that risks. In the spiral and rapid application development testing environment there may be no final functional requirements for the system. They are probably informal and evolutionary. In the spiral development environment, software testing is again described as a continuous improvement process that must be integrated into a rapid application development methodology. Testing as an integrated function prevents development from proceeding without testing. Deming's continuous improvement process using the PDCA model will again be applied to the software testing process.

Starting at the centre, each turn around the spiral goes through several task regions:

- Determine the objectives, alternatives, and constraints on the new iteration.
- Evaluate alternatives and identify and resolve risk issues.
- Develop and verify the product for this iteration.
- Plan the next iteration.

Spiral Model:

As the name implies, spiral model follows an approach in which there are a number of cycles (or spirals) of all the sequential steps of the waterfall model. Once the initial cycle gets completed, a thorough analysis and review of the achieved product or output is performed. If it is not as per the specified requirements or expected standards, a second cycle follows, and so on. This methodology follows an iterative approach and is generally suited for large projects having complex and constantly changing requirements.

The methodology provides a framework for testing in this environment. The major steps include information gathering, test planning, test design, test development, test execution/evaluation, and preparing for the next spiral. It includes a set of tasks associated with each step or a checklist from which the testing organization can choose based on its needs. The spiral approach flushes out the system functionality. When this has been completed, it also provides for classical system testing, acceptance testing, and summary reports.

The purpose of software testing is to identify the differences between existing and expected conditions, i.e., to detect software defects. Testing identifies the requirements that have not been satisfied and the functions that have been impaired. The most commonly recognized test objective is to identify bugs, but this is a limited definition of the aim of testing. Not only must bugs be identified, but they must be put into a framework that enables testers to predict how the software will perform. In the spiral and rapid application development testing environment there may be no final functional requirements for the system. They are probably informal and evolutionary. Also, the test plan may not be completed until the system is released for production. The relatively long lead time to create test plans based on a good set of requirements specifications may not be available. Testing is an ongoing improvement process that occurs frequently as the system changes. The product evolves and is not static. The testing organization needs to get inside the development effort and work closely with development. Each new version needs to be tested as it becomes available. The approach is to first test the new enhancements or modified software to resolve defects reported in the previous spiral. If time permits, regression testing is then performed to assure that the rest of the system has not regressed.

In the spiral development environment, software testing is again described as a continuous improvement process that must be integrated into a rapid application development methodology. Testing as an integrated function prevents development from proceeding without testing. Deming’s continuous improvement process using the PDCA model will again be applied to the software testing process. Before the continuous improvement process begins, the testing function needs to perform a series of information-gathering planning steps to understand the development project objectives, current status, project plans, function specification, and risks.

Once this is completed, the formal Plan step of the continuous improvement process commences. A major step is to develop a software test plan. The test plan is the basis for accomplishing testing and should be considered an ongoing document, i.e., as the system changes, so does the plan. The outline of a good test plan includes an introduction, the overall plan, testing requirements, test procedures, and test plan details. These are further broken down into business functions, test scenarios and scripts, function/test matrix, expected results, test case checklists, discrepancy reports, required software, hardware, data, personnel, test schedule, test entry criteria, exit criteria, and summary reports. The more definitive a test plan is, the easier the plan step will be. If the system changes between the development of the test plan and when the tests are to be executed, the test plan should be updated accordingly.

The Do step of the continuous improvement process consists of test case design test development and test execution. This step describes how to design test cases and execute the tests included in the test plan. Design includes the functional tests, GUI tests, and fragment system, and acceptance tests. Once an overall test design is completed, test development starts. This includes building test scripts and procedures to provide test case details.

The test team is responsible for executing the tests and must ensure that they are executed according to the test design. The do step also includes test setup, regression testing of old and new tests, and recording any defects discovered. The Check step of the continuous improvement process includes metric measurements and analysis. “Quality Through a Continuous Improvement Process,” crucial to the Deming method is the need to base decisions as much as possible on accurate and timely data. Metrics are key to verifying if the work effort and test schedule are on schedule, and to identify any new resource requirements. During the check step it is important to publish intermediate test reports. This includes recording of the test results and relating them to the test plan and test objectives. The Act step of the continuous improvement process involves preparation for the next spiral iteration. It entails refining the function/GUI tests, test suites, test cases, test scripts, and fragment system and acceptance tests, modifying the defect tracking system and the version and control system, if necessary. It also includes devising measures for appropriate actions relating to work that was not performed according to the plan or results that were not what was anticipated. Examples include a re-evaluation of the test team, test procedures, and technology dimensions of testing. All the above is fed back to the test plan, which is updated.

Once several testing spirals have been completed and the application has been verified as functionally stable, full system and acceptance testing starts. These tests are often optional. Respective system and acceptance test plans are developed defining the test objects and the specific tests to be completed. The final activity in the continuous improvement process is summarizing and reporting the spiral test results. A major test report should be written at the end of all testing. The process used for report writing is the same whether it is an interim or a final report, and, like other tasks in testing, report writing is also subject to quality control. However, the final test report should be much more comprehensive than the interim test reports. For each type of test it should describe a record of defects discovered, data reduction techniques, root cause analysis, the development of findings, and follow-on recommendations for the current and/or future projects.

The methodology provides a framework for testing in this environment. The major steps include information gathering, test planning, test design, test development, test execution/evaluation, and preparing for the next spiral. It includes a set of tasks associated with each step or a checklist from which the testing organization can choose based on its needs. The spiral approach flushes out the system functionality. When this has been completed, it also provides for classical system testing, acceptance testing, and summary reports.

Structural System Testing Techniques

Introduction :

Structural system testing aims to verify whether the developed system or program works correctly or not. This testing technique is implemented for ensuring accuracy/correctness of the design and functionality of the system. Structural system testing process is to determine that the technology used to build the product has been properly implemented. Also this technique aids in verifying that all components work as a single unit. The purpose of structural system testing is to ensure that it is designed correctly.

Structural system testing is thus executed with the help of various techniques.

Structural system Testing Techniques:

- **Stress Testing:**

  This testing primarily encompasses the test mechanisms used to validate the system's endurance capacity due to heavy amount of load. Data volume is checked on areas like transactions, disk space, output, memory, CPU utilizations etc. The idea behind stress testing is to determine how far the system's structure is able to handle large volumes of data.

Objectives of Stress Testing:

Generally, online systems should be tested by generating large volumes of data by involving as many users possible. When the system is checked, error conditions or the point at which an error occurs should be noted down.

  - Transactions for stress testing can be gathered from any of the three sources – test data generators, test transaction created by the test group and the transactions previously processed in the production environment.
  - This testing is a preferable choice when there is uncertainty regarding the amount of data to be handled by the application.

- **Execution Testing:**

  This type of testing aims to detect the performance levels in terms of response and turnaround times. Execution testing checks system's conformance with predefined performance criteria, that is, verifies optimum utilisation of software and hardware. This testing can be performed at any phase of the software development life cycle.

  Objectives of Execution testing:

  - To check the performance of the system's structure
  - To verify optimization of hardware and software
  - Check the response and turnaround time of user requests

- **Recovery Testing:**

  Well, at a certain point of time during testing, it may so happen that the system fails or stops responding. The state of the system is restored and transactions are reprocessed until the point of failure. Recovery testing simply ensures that the system is capable of restoring to the point where it failed. The motive is to preserve data in a secured location.

  Objectives of Recovery testing:

  - Determine that adequate backup data is maintained.
  - Ensure storing backup data in a secured location.
  - Develop recovery tools and make them available at the time of requirement.

- **Operations Testing:**

  After completing all the previous forms of testing, the product is integrated into the actual operating environment. At this stage, the application is tested under a real environment with normal staff, operations, procedures and documentation. The purpose is to ensure that the requirements are met and are complete and reasonable in the actual operational environment.

Objectives of Operations testing:

  - To determine completeness of documentation
  - To assess whether necessary support mechanisms are available.

- **Compliance Testing:**

  This testing helps to verify conformance of the application with the prescribed standards, procedures and guidelines. Compliance testing helps to forecast the success of the project and to enhance the chances of maintainability of the application system. This methodology helps the professionals working on the project comply with the standards so there are chances of least occurrence of errors.

  Objectives of compliance testing -

  - To ensure that standards are followed appropriate for the project.
  - Verify compliance to departmental standards, procedures and guidelines.

- **Security Testing:**

  Security testing aims at verifying whether integrity of confidential data is strictly maintained. Data protection is the most important part for an organisation as the vulnerability of data loss can hamper the business as a whole.

  Objectives of security testing:

  - To determine whether much needed attention is paid to identifying security risks.
  - To assess whether stringent measures have been implemented to enforce system's authorisation.
  - Determine whether an organisation has adequate number of people who are an adept at performing security testing.

Conclusion:

Structure or architecture of an application is the foundation upon which rest of the things depend. Structural testing can thus be implemented at all levels or at any phase throughout the software development life cycle.

FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements/specifications.

Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions.

During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester.

Functional testing is normally performed during the levels of System Testing and Acceptance Testing.

Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function’s specifications.
- Determine the output based on the function’s specifications.
- Execute the test case.
- Compare the actual and expected outputs.

Functional testing is more effective when the test conditions are created directly from user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users’ wrath when they finally use the software.

What is Functional Testing?

Functional Testing is the type of testing done against the business requirements of application. It is a black box type of testing.

It involves the complete integration system to evaluate the system’s compliance with its specified requirements. Based on the functional specification document this type of testing is to be carried out. In actual testing, testers need to verify a specific action or function of the code. For functional testing either manual testing or automation tools can be used but functionality testing would be easier using manual testing only. Prior to non Functional testing the Functional testing would be executed first.

Five steps need to be keeping in mind in the Functional testing:

1. Preparation of test data based on the specifications of functions
2. Business requirements are the inputs to functional testing

3. Based on functional specifications find out of output of the functions
4. The execution of test cases
5. Observe the actual and expected outputs

To carry out functional testing we have numerous tools available, here is the list of Functional testing tools.

- Unit Testing
- Smoke testing
- Sanity testing
- Integration Testing
- Interface Testing
- System Testing
- Regression Testing
- UAT

What is non-Functional Testing?

The non Functional Testing is the type of testing done against the **non functional requirements**. Most of the criteria are not consider in functional testing so it is used to **check the readiness of a system.** Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users. It can be started after the completion of Functional Testing. The **non functional tests** can be effective by using testing tools.

Non functional testing has a great influence on customer and user satisfaction with the product. Non functional testing should be expressed in a testable way, not like "the system should be fast" or "the system should be easy to operate" which is not testable. Basically in the non functional test is used to major **non-functional attributes of software systems.** Let's take **non functional requirements** examples; in how much time does the software will take to complete a task? or how fast the response is.

Following testing should consider in **non functional testing types**:

- Availability Testing
- Baseline testing
- Compatibility testing
- Compliance testing
- Configuration Testing

- Documentation testing
- Endurance testing
- Ergonomics Testing
- Interoperability Testing
- Installation Testing
- Load testing
- Localization testing and Internationalization testing
- Maintainability Testing
- Operational Readiness Testing
- Performance testing
- Recovery testing
- Reliability Testing
- Resilience testing
- Security testing
- Scalability testing
- Stress testing
- Usability testing
- Volume testing

Security Testing

SECURITY TESTING is a type of software testing that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders.

Focus Areas

There are four main focus areas to be considered in security testing (Especially for web sites/applications):

- **Network security:** This involves looking for vulnerabilities in the network infrastructure (resources and policies).
- **System software security:** This involves assessing weaknesses in the various software (operating system, database system, and other software) the application depends on.
- **Client-side application security:** This deals with ensuring that the client (browser or any such tool) cannot be manipulated.
- **Server-side application security:** This involves making sure that the server code and its technologies are robust enough to fend off any intrusion.

Example

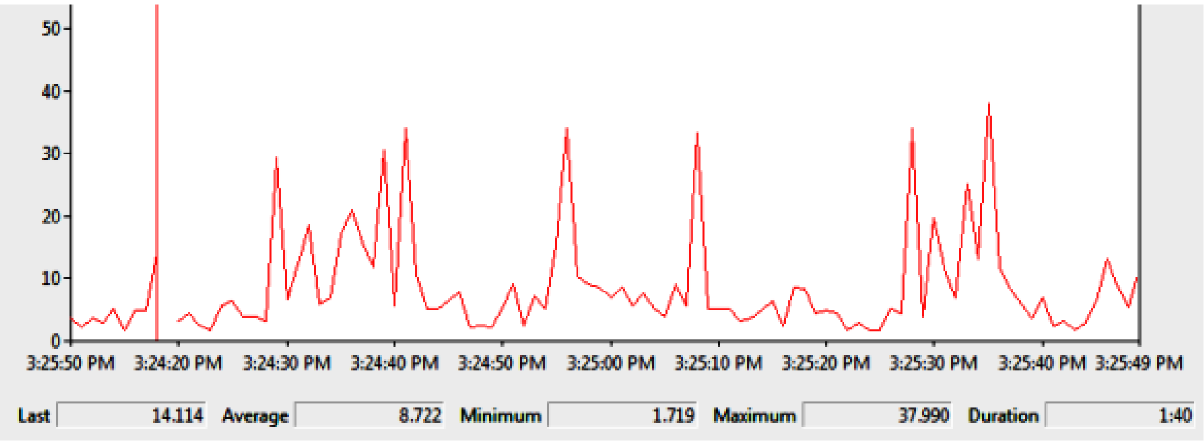
This is an example of a very basic security test which anyone can perform on a web site/application:

- Log into the web application.
- Log out of the web application.
- Click the BACK button of the browser (Check if you are asked to log in again or if you are provided the logged-in application.)

Most types of security testing involve complex steps and out-of-the-box thinking but, sometimes, it is simple tests like the one above that help expose the most severe security risks.

Performance Testing

PERFORMANCE TESTING is a type of software testing that intends to determine how a system performs in terms of responsiveness and stability under a certain load.



Last 14.114 Average 8.722 Minimum 1.719 Maximum 37.990 Duration 1:40

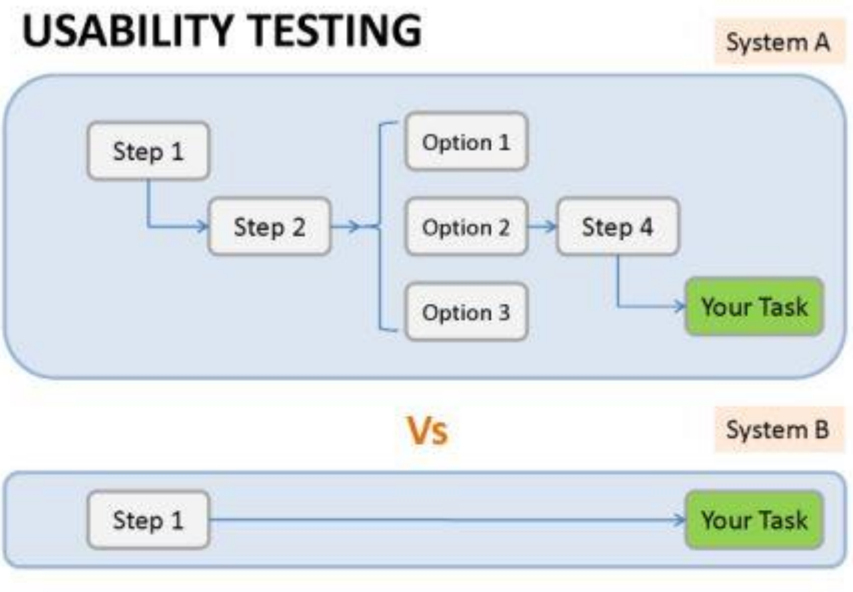
There are basically four kinds of performance testing:

Types

- **Load Testing** is a type of performance testing conducted to evaluate the behavior of a system at increasing workload.
- **Stress Testing** is a type of performance testing conducted to evaluate the behavior of a system at or beyond the limits of its anticipated workload.
- **Endurance Testing** is a type of performance testing conducted to evaluate the behavior of a system when a significant workload is given continuously.
- **Spike Testing** is a type of performance testing conducted to evaluate the behavior of a system when the load is suddenly and substantially increased.

Usability Testing

USABILITY TESTING is a type of testing done from an end-user’s perspective to determine if the system is easily usable.



Merriam-Webster’s Definition

usable

– capable of being used

– convenient and practicable for use

ISTQB’s Definition

usability testing: Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions.

CSTE CBOK Definition

Usability Test: The purpose of this event is to review the application user interface and other human factors of the application with the people who will be using the application. This is to ensure that the design (layout and sequence, etc.) enables the business functions to be executed as easily and intuitively as possible.

Elaboration

Systems may be built 100% in accordance with the specifications. Yet, they may be ‘unusable’ when it lands in the hands of the end-users. For instance, let’s say a user needs to print a Financial Update Report, every 30 minutes, and he/she has to go through the following steps:

1. Login to the system
2. Click Reports
3. From the groups of reports, select Financial Reports
4. From the list of financial reports, select Financial Update Report
5. Specify the following parameters
    1. Date Range
    2. Time Zone
    3. Departments
    4. Units
6. Click Generate Report
7. Click Print
8. Select an option
    1. Print as PDF
    2. Print for Real

If that’s the case, the system is probably practically unusable (though it functions perfectly fine). If the report is to be printed frequently, wouldn’t it be convenient if the user could get the job done in a couple of clicks, rather than having to go through numerous steps like listed above? What if there was a feature to save frequently generated reports as a template and if the saved reports were readily available for printing from the homepage?

What is Test Effectiveness?

Test Effectiveness can be defined as how effectively testing is done or goal is achieved that meets the customer requirement. In SDLC (Software development Life Cycle), we have requirements gathering phase where SRS (Software Requirements Specification) and FRD (Functional Requirements Document) are prepared and based on that development team starts building the software application, at the same time test cases are carved out of SRS and FRD documents by the testing team. Test effectiveness starts right at the beginning of the development and execution of test cases and after development is completed to count the number of defects. Defects can be valid or invalid. Valid defects are required to be fixed in the application or product and invalid ones need to be closed or ignored. Thus, mathematically it is calculated as a percentage of a number of valid defects fixed in software application divided by the sum of a total number of defects injected and a total number of defects escaped.

Test Effectiveness Formula:

Test Effectiveness = ((Defects removed in a phase) / (Defect injected + Defect escaped)) * 100

What is Test Efficiency?

Test Efficiency can be defined as the most efficient way in which the resources associated with the software project are utilized to achieve an organization goal (for example, number of projects to be completed in that particular year). Test efficiency is an internal process for an organization that evaluates the utilization of resources which could be a timeline, hardware, manpower, expertise of test team, etc. to develop a software product. Mathematically test efficiency is calculated as a percentage of the number of alpha testing (in-house or on-site) defects divided by sum of a number of alpha testing and a number of beta testing (off-site) defects. Alpha testing is the testing done by the expertise project test team on-site and this test team is expected to test the product thoroughly before the product is available to the customer or end user in the market. Once internal alpha testing is completed the product is made available to end users to test and look for the defects and provide their valuable feedback. Ideally, there should not be any defect observed by the end user during beta testing as any valid bug identified during beta testing, it will directly deteriorate the test efficiency of the on-site project team.

Factors Affecting Test Efficiency:

As explained above, the expectation from test team is 100% efficiency which is quite challenging. Onsite testing demands efficient project management, testing expert professionals, the best training to resources at technical as well as business level. Below are the key points to achieve almost 100% test efficiency.

- **Efficient Project Management:** Project management plays a key role in running the project smoothly. From a testing point of view, project management should make sure that the project has ample resources at headcount as well as expertise level. Project management should take care of the smooth transition from requirement phase to development and testing phase or in order words project requirements should be crystal clear. Project management should facilitate regular project meetings and track the project progress.
- **Expertise Test Professional:** In order to achieve 100% efficiency, one cannot deny on the fact that such test team demands testing professionals who are expert not only at a technical level but should have project domain knowledge and ability to think rationally on unforeseen test scenarios. For example, if the software product is related to the banking industry then a test professional having the knowledge of banking concepts suits best for this testing team.
    - **Project Training:** Not only best management and expertise test professional are sufficient to drive a software project efficiently but a software project demands regular project-related training to test professional so that they may sharpen their skills and contribute to the best testability of the software product.
    - **Technologies and Test automation:** Testing team should leverage the use of latest testing tools and technologies to automate the testing which will save the human effort and provide the ample time to the test team to test for various unforeseen test scenarios which can be later automated.