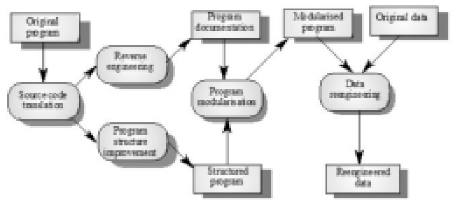
Software Re-Engineering

- Software Reengineering is a process of software development which is done to improve the maintainability of a software system.
  - It is the examination and alteration of a system to reconstitute it in a new form.
  - The principles of Re-Engineering when applied to the software development process is called software re-engineering.
  - It affects positively at software cost, quality, service to the customer and speed of delivery.
  - In Software Re-engineering, we are improving the software to make it more efficient and effective.

- Software Re-engineering is restructuring or rewriting part or all of a system without changing its functionality.
- It is applicable when some (but not all) subsystems of a larger system require frequent maintenance.

Software Re-Engineering …..

- Reengineering involves putting in the effort to make it easier to maintain.
- The reengineered system may also be restructured and should be re-documented
- When do you decide to reengineer?
  - When system changes are confined to one subsystem, the subsystem needs to be reengineered.
  - When hardware or software support becomes obsolete.
  - When tools to support restructuring are readily available.



Software Re-Engineering …..

- **Economics of Reengineering:**
  - Cost of maintenance:
    - Annual cost of operation and maintenance over application lifetime.
  - Cost of reengineering:
    - Predicted return on investment reduced by cost of implementing changes and engineering risk factors
  - Cost benefit:
    - Cost of re engineering - Cost of maintenance

- **Re-engineering advantages**:
  - Reduced risk.
  - There is a high risk in new software development. There may be development problems, staffing problems and specification problems.

- The complete Software Re-Engineering lifecycle includes:
  - Product Management:
    - Risks analysis, root cause analysis, business analysis, requirements elicitation and management, product planning and scoping, competitive analysis.

  - Research and Innovation:
    - Definition of a problem, data gathering and analysis, identifying a solution and developing best-of-breed or innovative algorithms, verification of quality for data and results, patent preparation.

Software Re-Engineering …..

  - Product Development:
    - Technology analysis and selection, software architecture and design, data architecture, deployment architecture, prototyping and production code development, comprehensive software testing, data quality testing, and product packaging and deployment preparation

  - Product Delivery and Support:
    - Hardware/Platform analysis and selection, deployment and release procedures definition, installations and upgrades, tracking support issues, organizing maintenance releases.

  - Project Management:
    - Brings efficiency and productivity to your software re-engineering project by utilizing modern, practical software project management, software quality assurance, data quality assurance, and advanced risk management techniques.

Software Reverse Engineering

- Reverse engineering is taking apart an object to see how it works in order to duplicate or enhance the object.
- The practice, taken from older industries, is now frequently used on computer hardware and software.
- Software reverse engineering involves reversing a program's machine code (the string of 0s and 1s that are sent to the logic processor) back into the source code that it was written in, using program language statements.

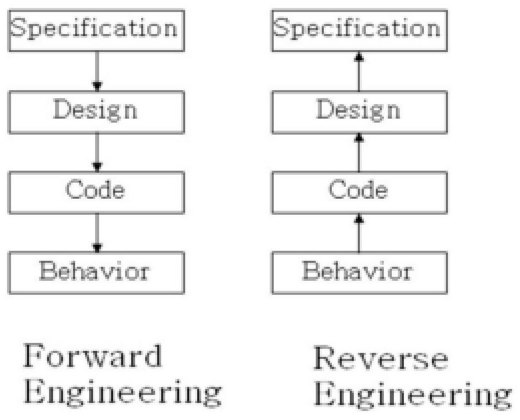
- Reverse-engineering is used for many purposes:
  - as a learning tool;
  - as a way to make new, compatible products that are cheaper than what i currently on the market;
  - for making software interoperate more effectively or to bridge data between different operating systems or databases; and
  - to uncover the undocumented features of commercial products.

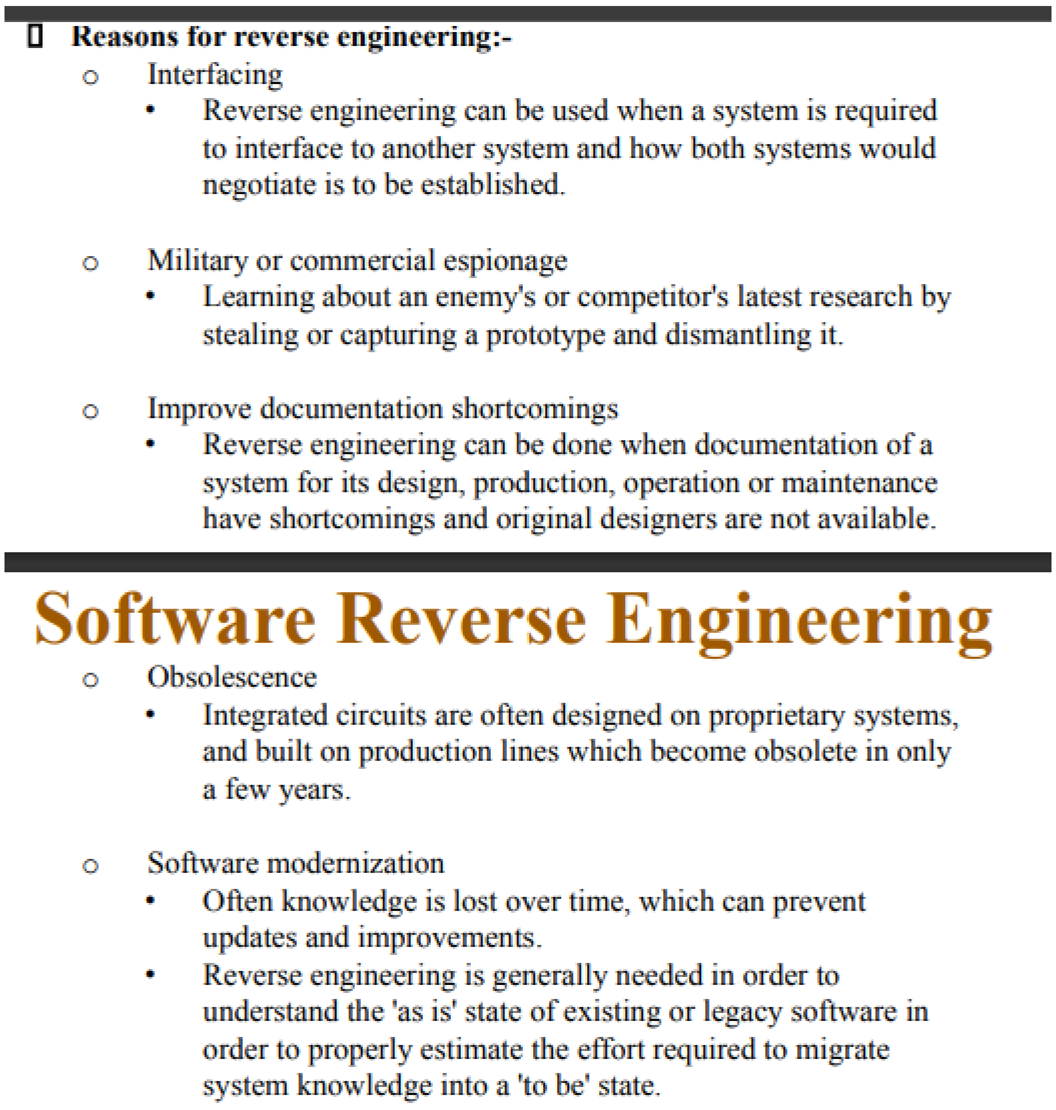
Software Reverse Engineering ….

- It is a process to achieve system specification by thoroughly analyzing, understanding the existing system.
  - This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels.
- An existing system is previously implemented design, about which we know nothing.
  - Designers then do reverse engineering by looking at the code and try to get the design.
- [illegible]

Software Reverse Engineering ….

- The usual reasons for reverse engineering a piece of software are to recreate the program, to build something similar to it, to exploit its weaknesses or strengthen its defenses.





- **Reasons for reverse engineering:-**
  - Interfacing
    - Reverse engineering can be used when a system is required to interface to another system and how both systems would negotiate is to be established.
  - Military or commercial espionage
    - Learning about an enemy's or competitor's latest research by stealing or capturing a prototype and dismantling it.
  - Improve documentation shortcomings
    - Reverse engineering can be done when documentation of a system for its design, production, operation or maintenance have shortcomings and original designers are not available.

Software Reverse Engineering

- Obsolescence
  - Integrated circuits are often designed on proprietary systems, and built on production lines which become obsolete in only a few years.
- Software modernization
  - Often knowledge is lost over time, which can prevent updates and improvements.
  - Reverse engineering is generally needed in order to understand the 'as is' state of existing or legacy software in order to properly estimate the effort required to migrate system knowledge into a 'to be' state.

Software maintenance

Software maintenance is a part of the Software Development Life Cycle. Its primary goal is to modify and update software application after delivery to correct errors and to improve performance. Software is a model of the real world. When the real-world changes, the software require alteration wherever possible.

Need for Maintenance

Software Maintenance is needed for:-

- Correct errors
- Change in user requirement with time

- Changing hardware/software requirements
- To improve system efficiency
- To optimize the code to run faster
- To modify the components
- To reduce any unwanted side effects.

Types of Software Maintenance

1. Corrective Maintenance

Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.

2. Adaptive Maintenance

It contains modifying the software to match changes in the ever-changing environment.

3. Preventive Maintenance

It is the process by which we prevent our system from being obsolete. It involves the concept of reengineering & reverse engineering in which an old system with old technology is re-engineered using new technology. This maintenance prevents the system from dying out.

4. Perfective Maintenance

It defines improving processing efficiency or performance or restricting the software to enhance changeability. This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.

CHP 06

What is Software Configuration Management?

- In Software Engineering, Software Configuration Management (SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Why do we need Configuration management?

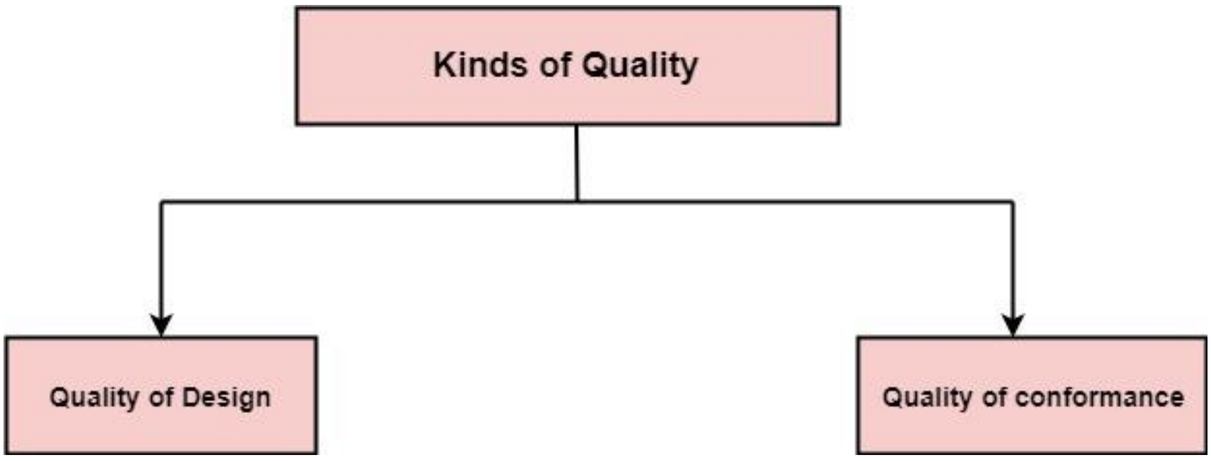
The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating
- It may be a case where multiple versions, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system.

What is Quality?

Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability.

There are two kinds of Quality:



Quality of Design: Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

Quality of conformance: Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.

Software Quality: Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

Quality Control: Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

Quality Assurance: Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.

Quality Assurance focuses on how the engineering and management activity will be done?

As anyone is interested in the quality of the final product, it should be assured that we are building the right product.

It can be assured only when we do inspection & review of intermediate products, if there are any bugs, then it is debugged. This quality can be enhanced.

SCM Process

It uses the tools which keep that the necessary change has been implemented adequately to the appropriate component. The SCM process defines a number of tasks:

- Identification of objects in the software configuration
- Version Control
- Change Control
- Configuration Audit
- Status Reporting
- **Identification**
- **Basic Object:** Unit of Text created by a software engineer during analysis, design, code, or test.
- **Aggregate Object:** A collection of essential objects and other aggregate objects. Design Specification is an aggregate object.
- Each object has a set of distinct characteristics that identify it uniquely: a name, a description, a list of resources, and a "realization."
- **The interrelationships between configuration objects can be described with a Module Interconnection Language (MIL).**
- **Version Control**
- Version Control combines procedures and tools to handle different version of configuration objects that are generated during the software process.
- **Clemm defines version control in the context of SCM:** Configuration management allows a user to specify the alternative configuration of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes.
- **Change Control**
- James Bach describes change control in the context of SCM is: Change Control is Vital. But the forces that make it essential also make it annoying.

- We worry about change because a small confusion in the code can create a big failure in the product. But it can also fix a significant failure or enable incredible new capabilities.
- We worry about change because a single rogue developer could sink the project, yet brilliant ideas originate in the mind of those rogues, and
- A burdensome change control process could effectively discourage them from doing creative work.
- A change request is submitted and calculated to assess technical merit; potential side effects, the overall impact on other configuration objects and system functions, and projected cost of the change.
- The results of the evaluations are presented as a change report, which is used by a change control authority (CCA) - a person or a group who makes a final decision on the status and priority of the change.
- The "check-in" and "check-out" process implements two necessary elements of change control-**access control** and **synchronization control**.
- **Access Control** governs which software engineers have the authority to access and modify a particular configuration object.
- **Synchronization Control** helps to ensure that parallel changes, performed by two different people, don't overwrite one another.
- **Configuration Audit**
- SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.
- SCM audits also ensure that traceability is maintained between all CIs and that all work requests are associated with one or more CI modification.
- SCM audits are the "**watchdogs**" that ensures that the integrity of the project's scope is preserved.
- **Status Reporting**
- Configuration Status reporting (sometimes also called status accounting) providing accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.

Risk management read ppt 1-30

CHP 03

Software Metrics

- A software metric is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.
- Within the software development process, many metrics are that are all connected. Software metrics are similar to the four functions of management: Planning, Organization, Control, or Improvement.

Classification of Software Metrics

Software metrics can be classified into two types as follows:

1. Product Metrics: These are the measures of various characteristics of the software product. The two important software characteristics are:

1. Size and complexity of software.
2. Quality and reliability of software.

These metrics can be computed for different stages of SDLC.

2. Process Metrics: These are the measures of various characteristics of the software development process. For example, the efficiency of fault detection. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software.

Types of Metrics:

A. **Internal metrics:** Internal metrics are the metrics used for measuring properties that are viewed to be of greater importance to a software developer. For example, Lines of Code (LOC) measure.
B. **External metrics:** External metrics are the metrics used for measuring properties that are viewed to be of greater importance to the user, e.g., portability, reliability, functionality, usability, etc.
C. **Hybrid metrics:** Hybrid metrics are the metrics that combine product, process, and resource metrics. For example, cost per FP where FP stands for Function Point Metric.

D. **Project metrics:** Project metrics are the metrics used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost; these estimates are used as a base of new software. Note that as the project proceeds, the project manager will check its progress from time-to-time and will compare the effort, cost, and time with the original effort, cost and time. Also understand that these metrics are used to decrease the development costs, time efforts and risks. The project quality can also be improved. As quality improves, the number of errors and time, as well as cost required, is also reduced.

Advantage of Software Metrics

- Comparative study of various design methodology of software systems.
- For analysis, comparison, and critical study of different programming language concerning their characteristics.
- In comparing and evaluating the capabilities and productivity of people involved in software development.
- In the preparation of software quality specifications.
- In the verification of compliance of software systems requirements and specifications.
- In making inference about the effort to be put in the design and development of the software systems.
- In getting an idea about the complexity of the code.
- In taking decisions regarding further division of a complex module is to be done or not.
- In guiding resource manager for their proper utilization.
- In comparison and making design trade-offs between software development and maintenance cost.

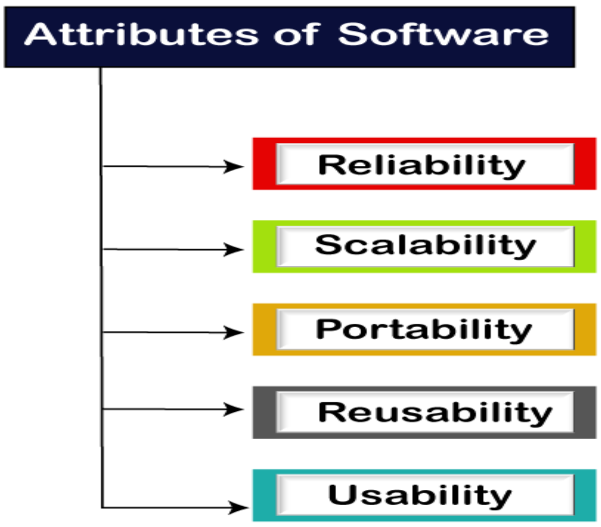
Disadvantage of Software Metrics

- The application of software metrics is not always easy, and in some cases, it is difficult and costly.
- The verification and justification of software metrics are based on historical/empirical data whose validity is difficult to verify.
- These are useful for managing software products but not for evaluating the performance of the technical staff.
- The definition and derivation of Software metrics are usually based on assuming which are not standardized and may depend upon tools available and working environment.

Testing

Software testing

Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.



- The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Need:

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Different Levels of Testing

The levels of software testing involve the different methodologies, which can be used while we are performing the software testing.

In software testing, we have four different levels of testing, which are as discussed below:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**

Unit Testing

- **Unit testing** is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.
- Unit is a process of validating such small building blocks of a complex
- system, much before testing an integrated large module or the system as a
- whole.
- It is a type of software testing where individual units or components of a
- software are tested.
- The purpose is to validate that each unit of the software code performs as
- expected.

Unit testing uses all white box testing techniques as it uses the code of software application:

- Data flow Testing
- Control Flow Testing
- Branch Coverage Testing
- Statement Coverage Testing
- Decision Coverage Testing

Advantages

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

Disadvantages

- It cannot identify integration or broad level error as it works on units of the code.

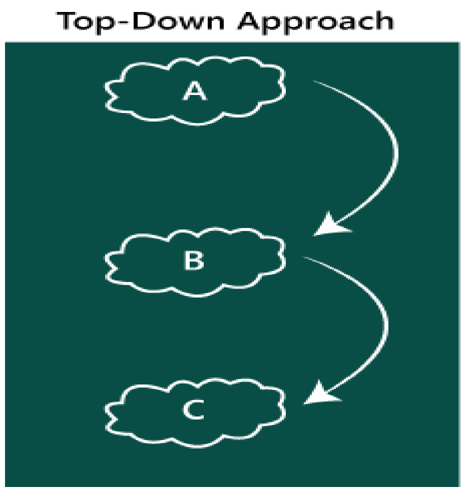
Integration Testing

- The second level of software testing is the **integration testing.** The integration testing process comes after **unit testing**.
- It is mainly used to test the **data flow from one module or component to other modules.**
- The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.
- When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.

. types:

Top-Down Approach

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules.



Advantages:

- Identification of defect is difficult.
- An early prototype is possible.

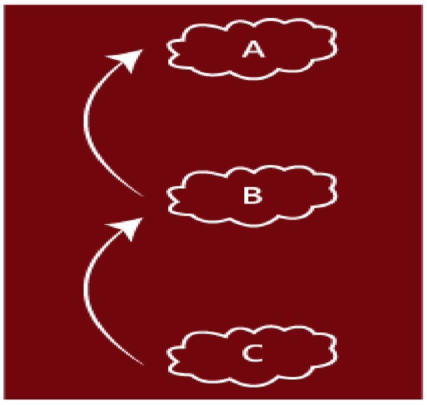
Disadvantages:

- Due to the high number of stubs, it gets quite complicated.
- Lower level modules are tested inadequately.

Bottom-Up Method

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules

Bottom-up Appoarch



Advantages

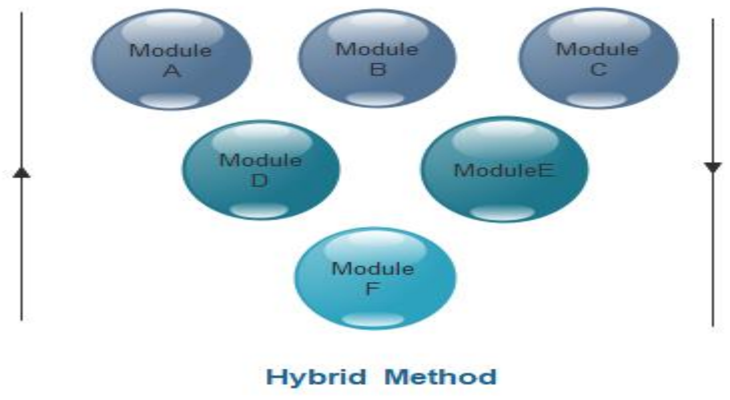
- Identification of defect is easy.
- Do not need to wait for the development of all the modules as it saves time.

Disadvantages

- Critical modules are tested last due to which the defects can occur.
- There is no possibility of an early prototype.

Hybrid Testing Method

In this approach, both **Top-Down** and **Bottom-Up** approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously



Advantages

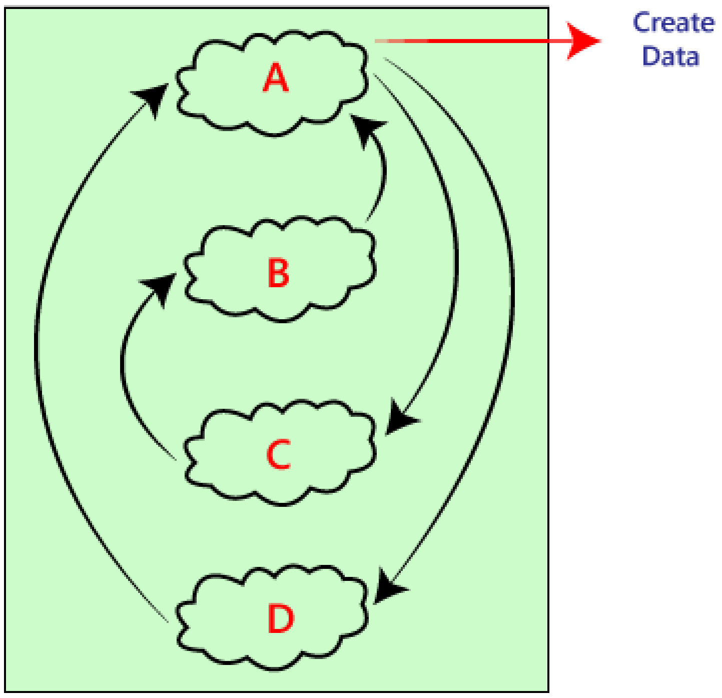
- The hybrid method provides features of both Bottom Up and Top Down methods.
- It is most time reducing method.
- It provides complete testing of all modules.

Disadvantages

- This method needs a higher level of concentration as the process carried out in both directions simultaneously.
- Complicated method.

Big Bang Method

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.



Advantages:

- It is convenient for small size software systems.

Disadvantages:

- Identification of defects is difficult because finding the error where it came from is a problem, and we don't know the source of the bug.
- Small modules missed easily.
- Time provided for testing is very less.

Level3: System Testing

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.

It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system.**

To check the end-to-end flow of an application or the software as a user is known as **System testing**.

Level4: Acceptance Testing

The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done.**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

S. No.	Black Box Testing	White Box Testing
1.	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	No knowledge of implementation is	Knowledge of implementation is

S. No.	Black Box Testing	White Box Testing
	needed.	required.
5.	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
6.	It is a functional test of the software.	It is a structural test of the software.
7.	This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
8.	No knowledge of programming is required.	It is mandatory to have knowledge of programming.
9.	It is the behavior testing of the software.	It is the logic testing of the software.
10.	It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
11.	It is also called closed testing.	It is also called as clear box testing.
12.	It is least time consuming.	It is most time consuming.
13.	It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
14.	Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
15.	**Example:** Search something on google by using keywords	**Example:** By input to check and verify loops
16.	**Black-box test design techniques-**	**White-box test design techniques-**

S. No.	Black Box Testing	White Box Testing
	• Decision table testing • All-pairs testing • Equivalence partitioning • Error guessing	• Control flow testing • Data flow testing • Branch testing
17.	**Types of Black Box Testing:** • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing
18.	It is less exhaustive as compared to white box testing.	It is comparatively more exhaustive than black box testing.