Câu hỏi môn Kiểm thử phần mềm

1. Trình bày Tài liệu đặc tả yêu cầu (SRS) các mục tiêu, chức năng chính của ứng dụng
2. Trình bày các nội dung chính có trong tài liệu TestPlan
3. Trình bày các mức trong qui trình kiểm thử phần mềm
4. Trình các kỹ thuật để tìm và thiết kế Test case
5. Trình bày các kỹ thuật kiểm thử hộp trắng (White Box Testing), khi nào áp dụng
6. Trình bày các kỹ thuật kiểm thử hộp đen (Blackbox testing), khi nào áp dụng
7. Giải thích các hoạt động trong qui trình phát triển phần mềm V model
8. Cho biết các loại kiểm thử nào sử dụng ở các mức nào
9. Nêu các độ đo trong kiểm thử phần mềm, độ phức tạp,
10. Cách thức để tiến hành viết Test Report và Defect list
11. Tại sao chúng ta cần thực hiện kiểm thử hộp trắng?
12. Phân biệt kiểm thử hộp trắng và kiểm thử hộp đen.
13. Tại sao kiểm thử hộp trắng thường có chi phí và độ khó cao hơn kiểm thử hộp đen?
14. Thế nào là đồ thị dòng điều khiển của một chương trình/đơn vị chương
15. Trình bày các độ đo kiểm thử cho kiểm thử dòng điều khiển.

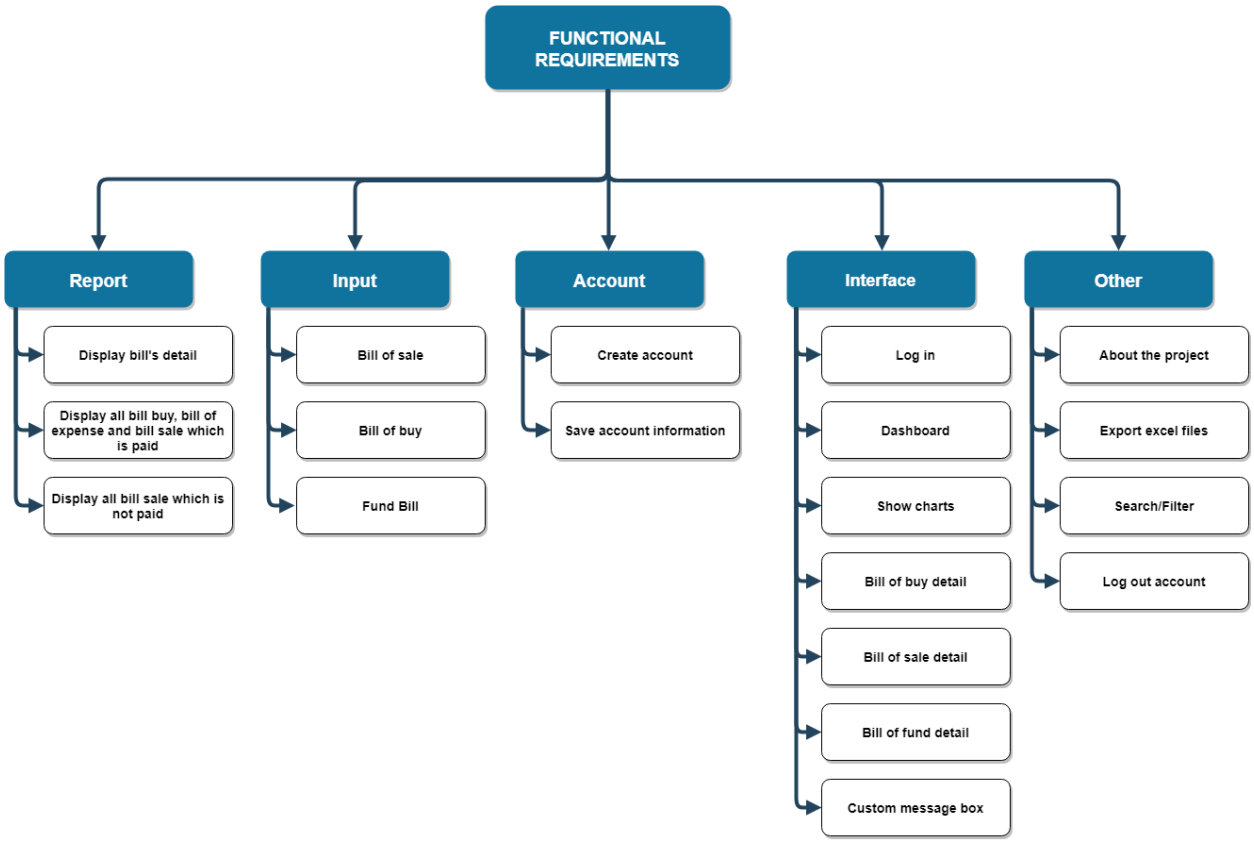
1.Tài liệu SRS

SRS là tài liệu được sử dụng để mô tả chi tiết các yêu cầu chức năng và phi chức năng của hệ thống. (Describe the detail of funtional and non-functional of the application)

Mục tiêu:

BAS is a business software that creates and manages business sales, purchase details in the easiest and most accurate way. To automate the business process of Point of Sale for Livel Bike. It should be user-friendly, fulfill all legal requirements and offer all functions needed in a wholesaler system.

Main function:



2. Test plan

In test planning, we establish (and update) the scope, approach, resources, schedule, and specific tasks in the intended test activities that comprise the rest of the test process. Test planning should identify test items, the features to be tested and not tested, the roles and responsibilities of the participants and stakeholders, the relationship between the testers and the developers of the test items, the extent to which testing is independent of development of these work items (see section 1.5 below), the test environments required, the appropriate test design techniques, entry, and exit criteria, and how we'll handle project risks related to testing.

3. Level of testing

There are mainly four **Levels of Testing** in software testing:

1. **Unit Testing** : checks if software components are fulfilling functionalities or not. test each module separately.

2. **Integration Testing** : checks the data flow from one module to other modules.

Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.

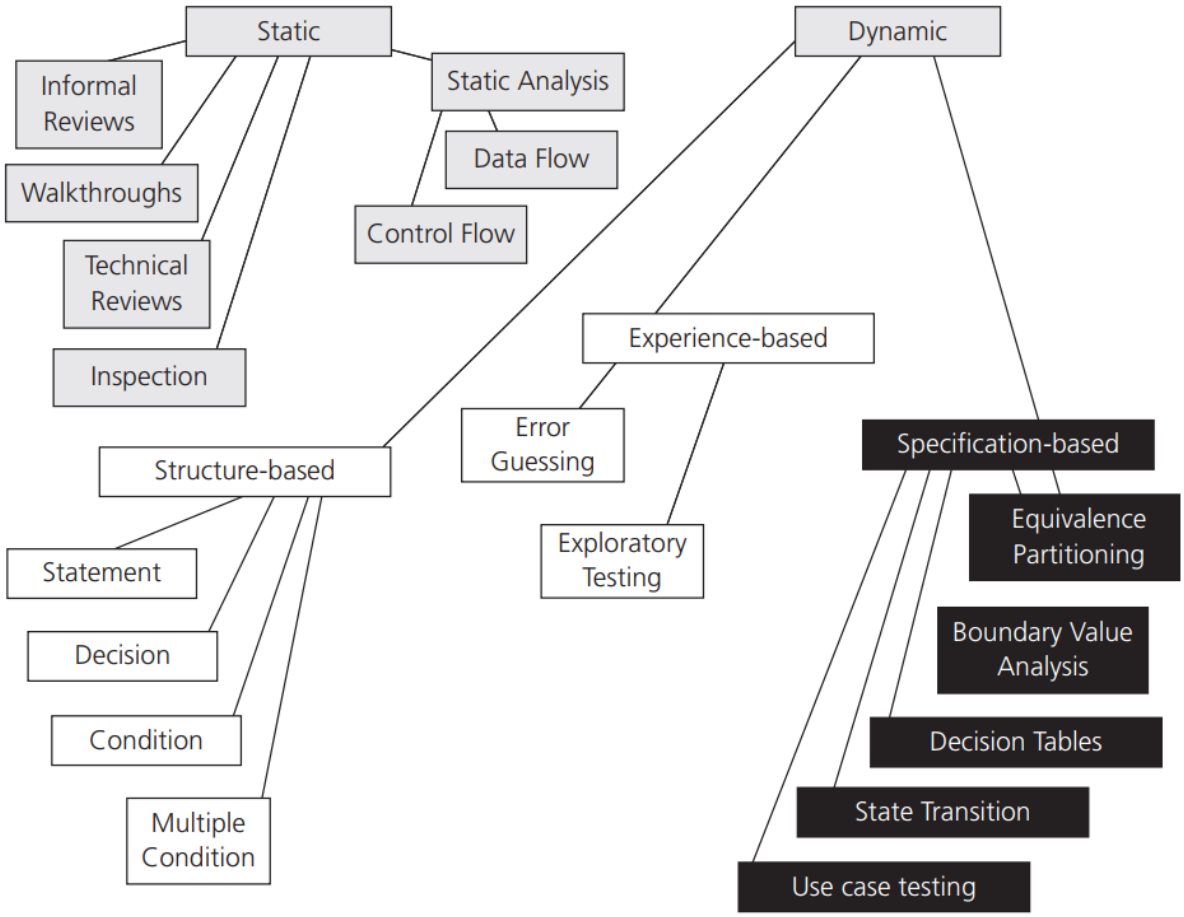
Integrating testing checks the data flow from one module to other modules

3. **System Testing** : evaluates both functional and non-functional needs for the testing.
4. **Acceptance Testing** : checks the requirements of a specification or contract are met as per its delivery

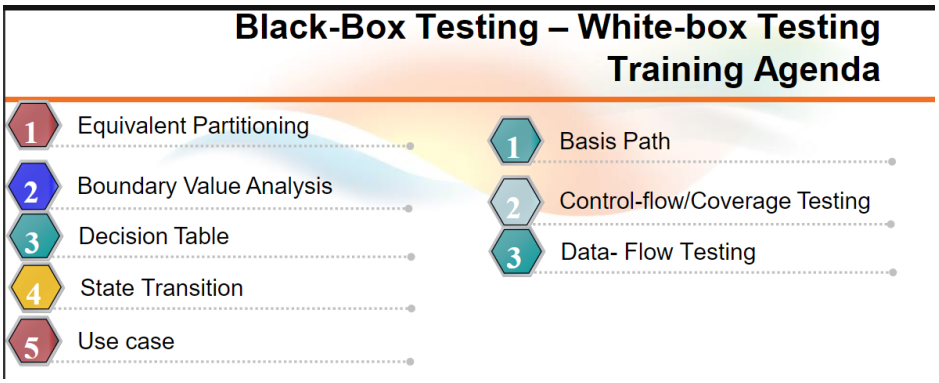
4. Kỹ thuật tìm và thiết kế testcase

- static testing techniques do not execute the code being examined and are generally used before any tests are executed on the software. Most static testing techniques can be used to 'test' any form of document including source code, design documents and models, functional specifications and requirement specifications

- Dynamic testing is an evaluation of that software or related work products that does involve executing the software. (Testing that involves the execution of the software of a component or system.) **Dynamic Testing** is a software testing method used to test the dynamic behaviour of software code. The main purpose of dynamic testing is to test software behaviour with dynamic variables or variables which are not constant and finding weak areas in software runtime environment



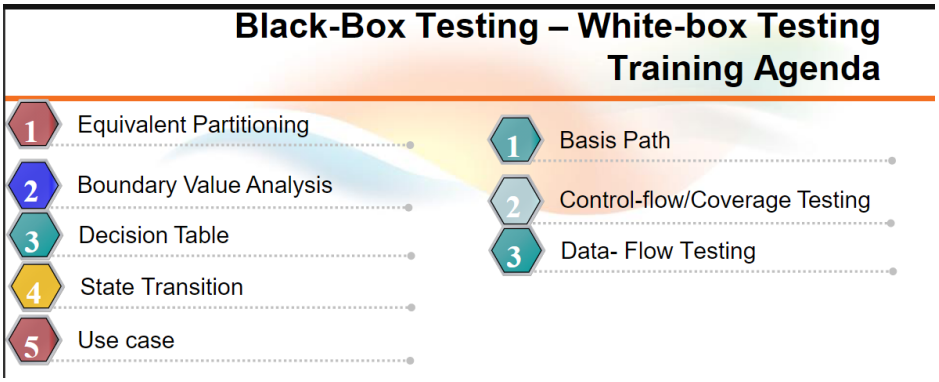
5. White box testing



- Structure-based testing techniques (which are also dynamic rather than static) use the internal structure of the software to derive test cases. They are commonly called ‘white-box’ or ‘glass-box’ techniques (implying you can see into the system) since they require knowledge of how the software is implemented, that is, how it works.

white-box techniques that would typically apply at the component level of testing. At this level, white-box techniques focus on the structure of a software component, such as statements, decisions, branches or even distinct paths. These techniques can also be applied at the integration level. At this level, white-box techniques can examine structures such as a call tree, which is a diagram that shows how modules call other modules. These techniques can also be applied at the system level. At this level, white-box techniques can examine structures such as a menu structure, business process or web page structure.

6. Black box



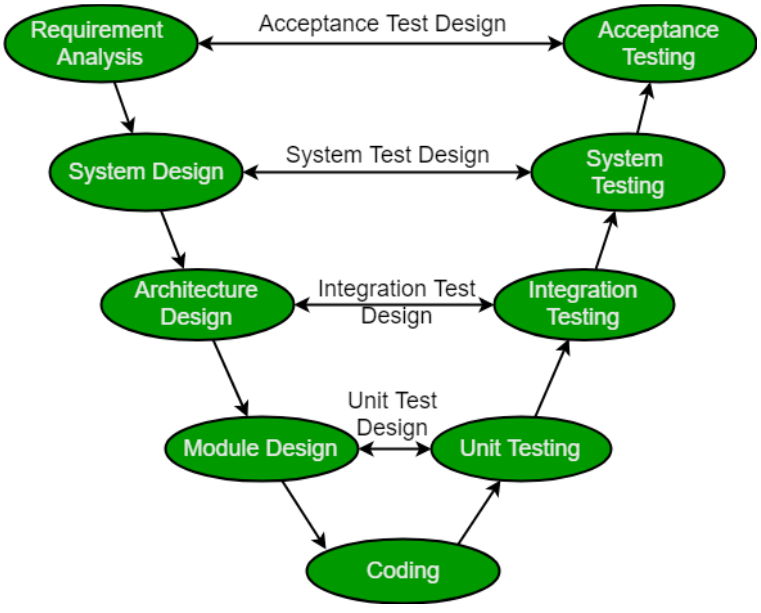
These are also known as ‘black-box’ or input/outputdriven testing techniques because they view the software as a black-box with inputs and outputs, but they have no knowledge of how the system or component is structured inside the box. In essence, the tester is concentrating on what the software does, not how it does it.

5. Vmodel

The V-model is a model that illustrates how testing activities (verification and validation) can be integrated into each phase of the life cycle.

Within the V-model, validation testing takes place especially during the early stages, e.g. reviewing the user requirements, and late in the life cycle, e.g. during user acceptance testing

Component testing: searches for defects in and verifies the functioning of
software components (e.g. modules, programs, objects, classes, etc.) that are
separately testable;
l **integration testing**: tests interfaces between components, interactions to
different parts of a system such as an operating system, file system and hardware
or interfaces between systems;
l **system testing**: concerned with the behaviour of the whole system/product as
defined by the scope of a development project or product. The main focus of
system testing is verification against specified requirements;
l **acceptance testing**: validation testing with respect to user needs, requirements, and
business processes conducted to determine whether or not to accept the system.



6. Loại kiểm thử dùng ở mức nào

WhiteBox

Will be used to cover some level of testing as unit testing, integration testing and system testing. However, it is almost used in unit testing

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before integration happens with previously tested code. White-box testing during unit testing potentially catches many defects early on and aids in addressing defects that happen later on after the code is integrated with the rest of the application and therefore reduces the impacts of errors later in development.[2]
2. Integration testing. White-box testing at this level is written to test the interactions of interfaces with each other. The unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.[2]

Blackbox

Black box testing can be used to test for functional and non-functional system requirements, though the majority of black box testing focuses on functional requirements.

Black box testing is used during Unit, Integration, System, and Acceptance testing.

In test types we can split out 4 kinds of types:

- Functional testing: Interoperability testing, Security testing

- Non-functional testing: Performance testing (load and stress), Usability testing, maintainability testing, reliability testing, portability testing
- Structural testing: Referred to as “white box testing” (Unit and integration test level)
- Confirm and regression testing: execute test case that have been tested before. (1 loại failed 1 loại passed)

Functional testing: Integration testing

Usability testing: System, acceptance testing

Interoperability testing

Security testing

Perfomance testing

+ Load testing

+ Stress testing

Usability testing: Acceptance testing

Regression testing: Can be at any test level, especially at system testing.

7. Cyclomatic complexity

Cyclomatic complexity: The number of independent paths through a program. Cyclomatic complexity is defined as: L – N + 2P, where
- L = the number of edges/ links in a graph
- N = the number of nodes in a graph

- P = the number of disconnected parts of the graph (e.g. a called graph or subroutine).

The cyclomatic complexity metric is based on the number of decisions in a
program. It is important to testers because it provides an indication of the amount
of testing (including reviews) necessary to detect a sufficient number of defects

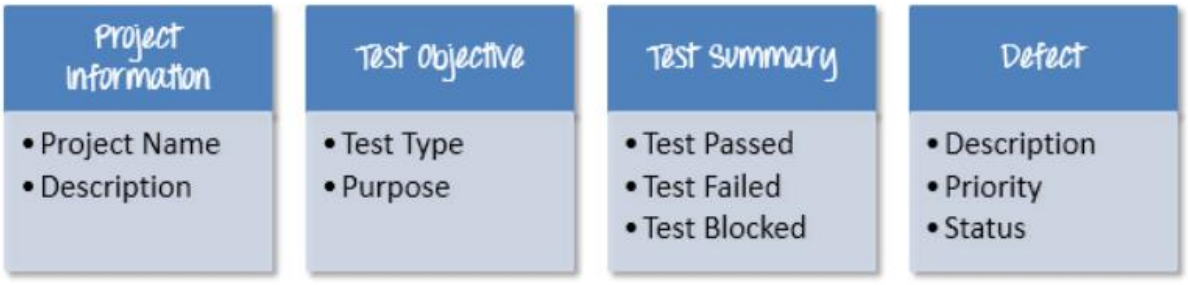
$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

Test coverage can be measured based on a number of different structural elements in a system or component. Coverage can be measured at component-testing level, integration-testing level or at system- or acceptance-testing levels.

For example, at system or acceptance level, the coverage items may be requirements, menu options, screens, or typical business transactions. Other coverage measures include things such as database structural elements (records, fields and sub-fields) and files.

8. Test report and Defect lists

Test reports contain:



Project Information

All information of the project such as the project name, product name, and version should be described in the test report. For example, the information of Guru99Bank project will be as follows

Project Overview				
PROJECT BASIC INFORMATION				
Project Name	Guru99 Bank			
Name of product (Product Number)	Banking website www.demo.guru99.com			
Product Description	The banking website			
Project Description	<**Mision of project**> Conduct testing to verify the quality of this website Ensure the website is released without any defects <**Project's output product**> Test Summary Report & Evaluation			
	Project Type	Testing/Verification		
Project Duration	Start date	10/1/2013	End date	10/31/2013

Test Objective

As mentioned in Test Planning tutorial, Test Report should include the objective of each round of testing, such as Unit Test, Performance Test, System Test …Etc.

Test Summary

This section includes the summary of testing activity in general. Information detailed here includes

- The number of test cases executed
- The numbers of test cases pass
- The numbers of test cases fail
- Pass percentage
- Fail percentage
- Comments

This information should be displayed **visually** by using **color indicator**, **graph, and highlighted table**.

Defect list

- Xác định lỗi (lỗi xảy ra ở chức năng nào?)
- Mô tả lỗi (Expected result và Relity Output)
- Xác định loại lỗi, mức độ ưu tiên, trạng thái

Defects

- This section should contain:
- Total number of detected bugs
- Bugs statuses (open, closed, fixed etc.)
- Number of bugs by the each status (open, closed, fixed etc.)
- Severity and priority breakdowns

Defect list:

Step 1: Define the defect

The first step is to define the defect by writing a summary in the defect title and providing a general description of the problem. When writing a summary in the defect title, include the area and function where the problem occurs

Step 2: Research the root cause

Research means making sure the defect is truly a defect. You'll want to check configuration settings, patient settings, user settings—anything in the application that affects how it functions.

Step 3: Add supporting documentation

Add or attach a step recorder file or video of the defect wherever possible.

Step 4: Format your report for high readability

Providing an understandable format makes your defect easier to review and more likely to be accepted. Format the defect text by separating it into the following sections:

- Summary (title)
- Description
- Build/platform
- Steps to reproduce
- Expected results
- Actual results
- Research
- Support documentation

9. Why we need White box

- It allows a finding of hidden errors, to find internal errors because it checks and works by internal functionality.
- It helps to find issues and optimize code to adopt different techniques of White Box Testing to test a developed application or website.

10. So sánh white and black

Tiêu chuẩn	Black box test	White box testing
1. Định nghĩa	- Kiểm tra hộp đen là phương pháp thử nghiệm phần mềm được sử dụng để kiểm tra các	- Kiểm tra hộp trắng là phương pháp kiểm thử phần mềm, sử dụng để kiểm tra

	phần mềm mà không quan tâm đến cấu trúc bên trong của chương trình.	phần mềm mà yêu cầu phải biết cấu trúc bên trong của chương trình.
2. Trách nhiệm	- Thử nghiệm được thực hiện bên ngoài, không liên quan đến nhà phát triển phần mềm.	- Thông thường, các thử nghiệm được thực hiện bởi nhà phát triển phần mềm.
3. Cấp độ test sử dụng	- Thử nghiệm áp dụng ở cấp độ cao như: kiểm tra hệ thống (System test), kiểm tra chấp nhận (Acceptance test)	Thử nghiệm được áp dụng ở mức độ thấp hơn như thử nghiệm đơn vị (Unit Test), thử nghiệm hội nhập (Interation testing)
4. Biết lập trình	- Không yêu cầu hiểu biết về Lập trình	Yêu cầu hiểu biết nhất định về lập trình.
5. Biết việc thực hiện chương trình	- Không yêu cầu hiểu về cấu trúc bên trong chức năng, và không cần hiểu làm thế nào để có được chức năng đó	Yêu cầu hiểu cấu trúc bên trong chức năng được thực hiện như nào.
6. Cơ sở tạo Test Cases	- Kiểm tra hộp đen được bắt đầu dựa trên tài liệu yêu cầu kỹ thuật	Kiểm tra hộp trắng được bắt đầu dựa trên các tài liệu thiết kế chi tiết

Black Box Testing	White Box Testing
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

Black Box Testing	White Box Testing
	program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred as outer or external software testing.	It is the inner or the internal software testing.
It is functional test of the software.	It is structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.
It is least time consuming.	It is most time consuming.

Black Box Testing	White Box Testing
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: search something on google by using keywords	**Example:** by input to check and verify loops

11. Tại sao white box khó và tốn chi phí

Because white-box testing is more thorough it becomes very expensive in **time and cost** to conduct. Although unit tests alleviate this somewhat, there is an initial investment that must be done to write the unit tests. Also, this type of testing can scale badly with large applications. It becomes virtually impossible to test every branch of code.

Compared to black-box testing, white-box testing requires skilled testers with programming knowledge. This increases the cost and could mean that developers are pulled off of (**cannot**) developing new features.

12. Control Flow Graph

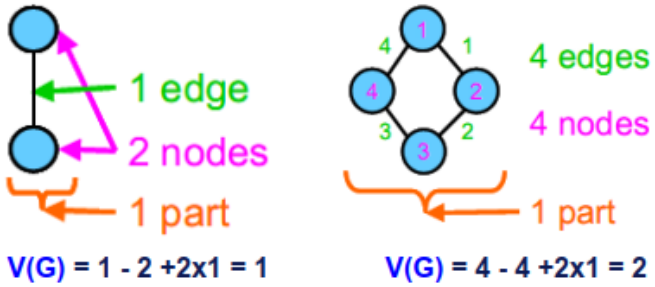
A Control Flow Graph (CFG) is the graphical representation of control flow or computation during the execution of programs or applications. Control flow graphs are mostly used in the static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit.

https://www.geeksforgeeks.org/software-engineering-control-flow-graph-cfg/

13. Cyclomatic

The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify.

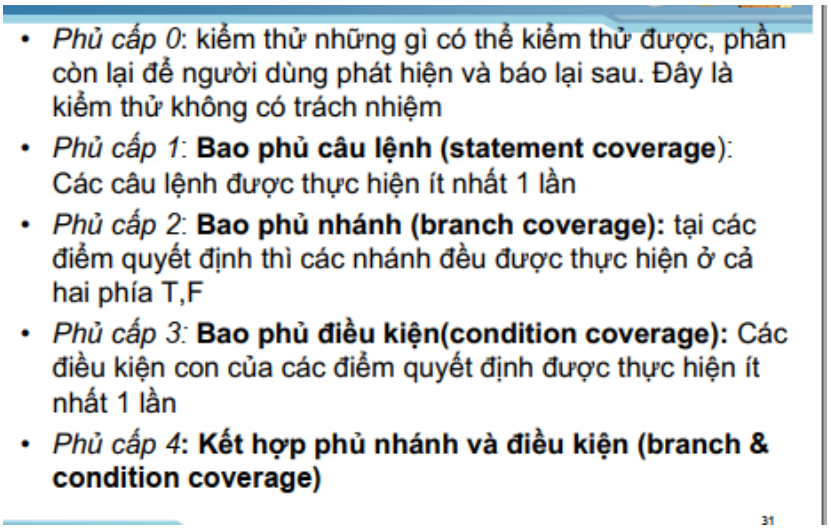
- Cách 1: Dựa trên công thức của McCabe
  - V(G) = edges - nodes + 2p
  - p = number of unconnected parts of the graph



Cyclomatic: Độ phức tạp chu trình

Các cấp bao phủ kiểm thử

- **Phủ kiểm thử (coverage)**: tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể các thành phần.
- Các thành phần bao gồm: lệnh thực thi, điểm quyết định, điều kiện con hay sự kết hợp của chúng.
- Độ phủ càng lớn thí độ tin cậy càng cao.



Question from Discussion

14. What is exhaustive testing?

Exhaustive testing is a test approach in which the test suite comprises all combinations of input values and preconditions.

15. What are testing principles?

Here are the 7 Principles:

- Exhaustive testing is not possible
  ⇨ instead, we need the optimal amount of testing based on the risk assessment of the application.
- Defect Clustering
  ⇨ Defect Clustering which states that a small number of modules contain most of the defects detected.
- Pesticide Paradox
  ⇨ Repetitive use of the same pesticide mixes to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of

pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

- Testing shows a presence of defects
  ⇨ Hence, testing principle states that – Testing talks about the presence of defects and don't talk about the absence of defects.
- Absence of Error – fallacy
  ⇨ It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs.
- Early Testing
  ⇨ Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages.
- Testing is context dependent
  ⇨ Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application

16. Why is testing necessary?

The testing is important since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software.

It makes the software more reliable and easy to use.

Thoroughly tested software ensures reliable and high-performance software operation.

17. What is testing?

Testing is part of how these risks of failure can be reduced

18. What is bug?

A software bug can also be issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.

19. What is fault?

Fault is an incorrect step in any process and data definition in computer program which is responsible of the unintended behavior of any program in the computer

20. How does testing influence and quality and other factors?

- Testing is part of how these risks of failure can be reduced
- Testing does not change the quality of the system under test directly. When testing finds defects and those defects are repaired, the quality of the system is increased.
- Testing does provide a way of measuring the system's quality
- Testing also provides a learning opportunity that allows for improved quality if lessons are learned from each project

21. What is test specification?

Test specification is a detailed summary of what scenarios will be tested, how they will be tested, how often they will be tested, and so on and so forth, for a given feature.

22. What is V-model?

A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model

illustrates how testing activities can be integrated into each phase of the software development lifecycle.

23. What are different between Static testing and dynamic testing techniques

	Static Testing	Dynamic Testing
2	It is performed in the early stage of the software development.	It is performed at the later stage of the software development.
3	In static testing whole code is not executed.	In dynamic testing whole code is executed.
4	Static testing prevents the defects.	Dynamic testing finds and fixes the defects.
5	Static testing is performed before code deployment.	Dynamic testing is performed after code deployment.
6	Static testing is less costly.	Dynamic testing is highly costly.
7	Static Testing involves checklist for testing process.	Dynamic Testing involves test cases for testing process.
8	It includes walkthroughs, code review, inspection etc.	It involves functional and nonfunctional testing.
9	It generally takes shorter time.	It usually takes longer time as it involves running several test cases.
10	It can discover variety of bugs.	It expose the bugs that are explorable through execution hence discover only limited type of bugs.
11	Static Testing may complete 100% statement coverage in comparably less time.	While dynamic testing only achieves less than 50% statement coverage.

24. Which steps are the review process?

6 steps
1 Planning
2 Kick-off
3 Preparation
4 Review meeting
5 Rework
6 Follow-up

25. What is the main difference between a walkthrough and an inspection?

A walkthrough is lead by the author, whilst an inspection is lead by a trained moderator

S.No.	Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by project team.	Initiated by author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in the walkthrough.
5.	Inspection processes includes overview, preparation, inspection, and rework and follow up.	Walkthrough process includes overview, little or no preparation, little or no preparation examination (actual walkthrough meeting), and rework and follow up.
6.	Formalized procedure in each step.	No formalized procedure in the steps.
7.	Inspection takes longer time as list of items in checklist is tracked to completion.	Shorter time is spent on walkthrough as there is no formal checklist used to evaluate program.
8.	Planned meeting with the fixed roles assigned to all the members involved.	Unplanned
9.	Reader reads product code. Everyone inspects it and comes up with detects.	Author reads product code and his teammate comes up with the defects or suggestions.
10.	Recorder records the defects.	Author make a note of defects and suggestions offered by teammate.
11.	Moderator has a role that moderator making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.

26. What are roles in review meeting?

moderator, author, scribe and reviewer, manager

1. **Moderator :**
   Moderator, also known as review leader, generally leads review process. It simply co-ordinates with author and checks entry criteria for review. During review, they also lead discussion.
2. **Author :**
   As the writer of ‘document under review’, author’s main goal must be to learn and know as much as possible with regard to improving and increasing quality of document. He/She is writer of documents under review and aims to gain maximum from review. They also help in improving document quality.

3. **Scribe/Recorder :**
   As the name suggests, scribe or recorder simply has to record each and every defect and error that is found and any suggestions or feedback given in the meeting for improvement of process. Basically, author plays role of scribe, but it's very good and advantageous to have another person to scribe so that author can only concentrate on review.
4. **Reviewer :**
   Reviewers only have to check all of the defects and errors and then further improvements also in accordance to business specifications, standards, and domain knowledge. Reviewers must be technical, sometimes when review documents are very few than highly skilled reviewers are needed.
5. **Manager :**
   Managers are basically included and involved in the reviews as he/she simply decides on execution of reviews, allocates time in project schedules, and generally identifies whether or not objectives of review project have been met. They decide execution of reviews. They also play role of reviewers sometimes.

27. What is Estimating testing?

Test Estimation is a management activity which approximates **how long** a Task would take to complete. Estimating effort for the test is one of the **major** and **important** tasks in Test Management.

28. What are differences between EP and BVA?

S.NO.	Boundary value analysis	Equivalence partitioning
1.	It is a technique where we identify the errors at the boundaries of input data to discover those errors in the input center.	It is a technique where the input data is divided into partitions of valid and invalid values.
2.	Boundary values are those that contain the upper and lower limit of a variable.	In this, the inputs to the software or the application are separated into groups expected to show similar behavior.
3.	Boundary value analysis is testing the boundaries between partitions.	It allows us to divide a set of test conditions into a partition that should be considered the same.
4.	It will help decrease testing time due to a lesser number of test cases from infinite to finite.	The Equivalence partitioning will reduce the number of test cases to a finite list of testable test cases covering maximum possibilities.
5.	The Boundary Value Analysis is often called a part of the Stress and Negative Testing.	The Equivalence partitioning can be suitable for all the software testing levels such as unit, integration, system.
6.	Sometimes the boundary value analysis is also known as Range Checking.	Equivalence partitioning is also known as Equivalence class partitioning.

29. What and how to set Entry and exit Criteria?

Entry criteria can be defined as specific conditions; or, all those documents which are required to start a particular phase of STLC should be present before entering any of the STLC phase.

An example of entry criteria can be as follows.

- *All the requirements are finalized and approved by the BA team.*
- *Requirement document is ready.*
- *All the defects from previous testing stages are closed.*
- *Dev sanity is completed.*

Exit criteria can be defined as items/documents/actions/tasks that must be completed before concluding the current phase and moving on to the next phase.

Exit criteria can depend on many factors such as time, budget etc.

The entry criteria should include the completion of exit criteria of the previous phase. In real time, it is not possible to wait for the next phase until the exit criteria is met. The next phase can be initiated if the critical deliverables of the previous phase have been completed.

30. What is risk in testing?

Risks in software testing are the possible problems that might endanger the objectives of the project stakeholders. It is the possibility of a negative or undesirable outcome. A risk is something that has not happened yet and it may never happen; it is a potential problem.

31. What is Incident management?

Incident management is basically an art of identifying, investigating and taking necessary actions to prevent such an event. The idea behind incident management is to ensure that incidents are tracked from its identification stage until its correction stage, so that the final result is bug free. Roles and responsibilities should be assigned to monitor the whole process of incident management.

Cái này ngắn hơn và từ ebook ra
Incident management*: The process of recognizing, investigating, taking action and disposing of incidents. It involves logging incidents, classifying them and identifying the impact.*