JUNAID . GIRKAR
6000419005?
TE COMPS A4

SOFTWARE ENGINEERING
ASSIGNMENT - 2

Q1 Explain equivalence partitioning and boundary value analysis

ANS EQUIVALENCE PARTITIONING :

- It is a black-box testing method that divides the input domain of a program into classes of data from which tests can be derived.
- An ideal test case single-handedly uncovers a class of errors (e.g. incorrect processing of all character data) that might otherwise require many test cases to be executed before the general error is observed.
- Test case design for equivalence partitioning is based on an evaluation of equivalence classes for an input condition.
- If a set of objects can be linked by relationships that are symmetric, transitive and reflexive, an equivalence class is present.
- An equivalence class represents a set of valid or invalid states for input conditions.
- Typically, an input condition is either a specific numeric value, a range of values, a set of related values, or a Boolean condition.
- Equivalence classes may be defined according to the following guidelines :-
  - If an input condition specifies a range, one valid and two invalid equivalence classes are defined.

- If an input condition specifies a member of a set, one valid and one invalid equivalence class are defined
- If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
- If an input condition is Boolean, one valid and one invalid class are defined.

- By applying the guidelines for the derivation of equivalence classes, test cases for each input domain data item can be developed and executed.
- Test cases are selected so that the largest number of attributes of an equivalence class are exercised at once.

BOUNDARY VALUE ANALYSIS

- A great number of errors occur at the boundaries of the input domain rather than in the "center".
- It is for this reason that boundary value analysis (BVA) has been developed as a testing technique.
- Boundary value analysis leads to a selection of test cases that exercise bounding values.
- Boundary value analysis is a test-case design technique that complements equivalence partitioning.
- Rather than selecting any element of an equivalence class, BVA leads to the selection of test cases at the "edges" of the class.
- Rather than focussing solely on input conditions, BVA derives test cases from the output domain as well.
- Guidelines for BVA are similar in many respects to those provided for equivalence partitioning :

1. If an input condition specifies a range bounded by values a and b, test cases should be designed with values just above and just below a and b.
2. If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and just below the minimum and maximum are also tested.
3. Apply guidelines 1 and 2 to output conditions. For example, assume that a temperature versus pressure table is required as output from an engineering analysis program. Test cases should be designed to create an output report that produces the maximum and minimum allowable number of entries in the table.
4. If internal program data structures have prescribed boundaries (e.g. a table has a defined limit of 100 entries), be certain to design a test case to exercise the data structure at its boundary.

Most software engineers perform BVA intuitively to some degree. By applying these guidelines, boundary testing will be more complete, thereby having a higher likelihood for error detection.

Q2. With suitable example, explain OAT

ANS:
- Orthagonal Array Testing (OAT) can be applied to problems in which the input domain is relatively small but too large to accomodate enhaustive testing.
- OAT is particularly useful in finding region faults — an error category associated with faulty logic within a software component.
- OAT strategy is one of the test case optimization techniques.
- Orthagonal Testing is a Black Box testing - test cases optimization technique used when the system to be tested has huge data inputs.
- For example : when a train ticket has to be verified, factors such as - the number of passengers, ticket number, seat numbers and the train numbers has to be tested, which becomes difficult when a tester verifies input one by one.
- Hence it will be more efficient when we combine more inputs together and test. Here we use OAT.
- This type of pairing or combining of inputs and testing the system to save time is called Pairwise testing. OAT's technique is used for pairwise testing.

▸ OAT advantages :-

→ Guarantees testing of the pair-wise combinations of all the selected variables.

→ Reduces the number of test cases.

→ Creates fewer test cases which cover the testing of all the combinations of all variables

→ A complex combination of the variables can be done.
→ It is simpler to generate less error-prone tests than test sets created by hand.
→ It is useful for integration testing.
→ It improves productivity due to reduced test cycles and testing times.

▸ OAT DISADVANTAGES :-
→ As data inputs increase, the complexity of test cases increases. As a result, manual effort and time spent increases. Hence the testers have to go for automation testing.
→ useful for integration testing of software components

OAT EXAMPLE

- A web page has three distinct sections (Top, Middle and Bottom) that can be individually shown to or hidden from user
- Number of factors = 3 [Top, Middle & Bottom]
- Number of levels (visibility) = 2 [Hidden or Shown]
- Conventional Testing : 2 x 3 = 6 test cases.
- OAT Testing : 4 test cases.

CONVENTIONAL TESTING

Test cases	Scenarios	values to be tested
# 1	Hidden	Top
# 2	Shown	Top
# 3	Hidden	Bottom
# 4	Shown	Bottom
# 5	Hidden	Middle
#6	Shown	Middle

OAT

Test cases	TOP	MIDDLE	BOTTOM
# 1	Hidden	Hidden	Hidden
# 2	Hidden	Visible	Visible
# 3	Visible	Hidden	Visible
# 4	Visible	Visible	Hidden

Q3 Explain version control in SCM

ANS
- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process.
- A version control system implements or is directly integrated with four major capabilities :-

(1) A project database (repository) that stores all relevent configuration objects.

(2) A version management capability that stores all relevent configuration objects (or enables any version to be constructed using differences from past versions)

(3) A 'make facility' that enables us to collect all relevant configuration objects and construct a specific version of the software.

- In addition, version control and change control systems often implement an issue tracking (also called bug tracking) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.
- A number of version control systems establish a change set - a collection of all changes (to some baseline configuration) that are required to create a specific version of the software.
- A number of named change sets can be identified for an application or system. This enables us to construct a version of the software by specifying the change sets (by name) that must be applied to the baseline configuration.

• To accomplish this, a system modeling approach is applied. The system model contains :

(1) A template that includes a component hierarchy and a "build order" for the components that describe how the system must be constructed.

(2) construction rules

(3) Verification rules

• A number of different automated approaches to version control have been proposed over the last few decades

• The primary difference in approaches is the sophistication of the attributes that are used to construct specific versions and variants of a system and the mechanics of the process for construction.

→ Need for version control in software configuration management

• VERSIONING : As a project progresses, many versions of individual work products will be created. The repository must be able to save all of these versions to enable efficient management of product releases and to permit developers to go back to previous versions during testing and debugging.

• Dependency tracking and change management : The repository manages a wide variety of relationships among the data elements stored in it. The ability to keep track of all these relationships is crucial to the integrity of the information stored in the

repository and to the generation of deliverables based on it.

- REQUIREMENTS TRACING : Depends on link management and provides the ability to track all the design and construct components and deliverables that result from a specific requirements specification (forward tracing). It also identifies which requirement generates any given work product (backward tracing)

- CONFIGURATION MANAGEMENT : Keeps track of a series of configurations representing specific project milestones or production releases

- AUDIT TRAILS : Establish additional information about when, why, and by whom changes are made.

Examples of version control softwares are Git, Github, GitLab, Bit Bucket, etc.