OBJECT ORIENTED TESTING METHODS

[redacted]
[redacted]
Computer science and Engineering

CHAPTER-3

OBJECT ORIENTED TESTING METHODS

Topics to be Covered

- Applicability of Conventional Test Case Design Methods
- Issues in Object Oriented Testing,
- Fault-Based Testing
- Scenario-Based Testing
- Random Testing
- PartitionTestingfor Classes
- Interclass Test Case Design

Object Oriented Testing

- Find the greatest possible number of errors with a manageable amount of effort applied over a realistic time span.
- The nature of OO programs changes both testing strategy and testing tactics.
- Do we need less efforts for testing because of greater reuse of design patterns?
- Answer is no! Binder argues that more testing is needed to obtain high reliability in OO systems, since each reuse is a new context of usage.

Testing Object-Oriented Applications

Why its different

- No sequential procedural executions
- No functional decomposition
- No structure charts to design integration testing
- Iterative O-O development and its impact on testing and integration strategies

Objectives

To cover the strategies and tools associated with object oriented testing

- Analysis and Design Testing
- Unit/Class Tests
- Integration Tests
- Validation Tests
- System Tests

To discuss test plans and execution for projects

Test Strategies

Issues to address for a successful software testing strategy:

- Specify product requirements long before testing commences
  For example: portability, maintainability, usability
  Do so in a manner that is unambiguous and quantifiable
- Understand the users of the software, with use cases
- Develop a testing plan that emphasizes “rapid cycle testing”
  Get quick feedback from a series of small incremental tests
- Build robust software that is designed to test itself
  Use assertions, exception handling and automated testing tools (Junit).
- Conduct formal technical reviews to assess test strategy & test cases “Who watches the watchers?”

Testing OOA and OOD Models

- The review of OO analysis and design models is especially useful because the same semantic constructs (e.g., classes, attributes, operations, messages) appear at the analysis, design, and code level.
- Therefore, a problem in the definition of class attributes that is uncovered during analysis will circumvent side effects that might occur if the problem were not discovered until design or code (or even the next iteration of analysis).
- By fixing the number of attributes of a class during the first iteration of OOA, the following problems may be avoided:
  Creation of unnecessary subclasses.
  Incorrect class relationships.
  Improper behavior of the system or its classes.
- If the error is not uncovered during analysis and propagated further more efforts needed during design or coding stages.

Applicability of Conventional Test Case Design

•A collection of layered subsystems that encapsulate collaborating classes results in the design of object-oriented applications.

•Each of these components of the system (subsystems and classes) carries out functions that help to fulfil system requirements.

•In an attempt to discover errors, it is important to evaluate an OO system at a number of different levels.

•As classes collaborate with each other and subsystems connect through architectural layers, this can happen.

Applicability of Conventional Test Case Design

•OO testing is similar strategically to the testing of traditional systems, but it is distinct tactically.

•Since the OO research and design models are identical to the resulting OO software in structure and content, "testing" starts with.

•The interpretation of these models. OO testing starts "in the small" with class testing once the code has been created.

•When one class collaborates with other classes, a series of tests are designed to conduct class operations and investigate whether errors occur.

Applicability of Conventional Test Case Design

- With the development of research and design models, the construction of object-oriented software begins.
- The evolutionary essence of the model of OO software engineering is attributed to
- These models begin as relatively informal representations of the specifications of the system and grow into comprehensive class models.
- Connections and relationships of class, machine design and allocation, and design of objects.
- At each point, in an effort to uncover errors before their propagation to the next iteration, the models can be checked

Applicability of Conventional Test Case Design

- It can be argued that, because of the same semantic constructs, the study of OO analysis and design models is particularly helpful.
- A problem in the class attribute description that is discovered during the study will bypass side effects
- Effects that could occur if before design, the issue was not discovered.
- The class's testing will take more time than necessary.
- Once the problem is eventually discovered, the device must be updated.

Applicability of Conventional Test Case Design

1.Identify each test case uniquely
- Associate test case explicitly with the class and/or method to be tested

2.State the purpose of the test

3.Each test case should contain:

a. list of specified states for the object that is to be tested
b. A list of messages and operations that will be exercised as a consequence of the test
c. A list of exceptions that may occur as the object is tested
d. A list of external conditions for setup (i.e., changes in the environment external to the software that must exist in order to properly conduct the test)
e. Supplementary information that will aid in understanding or implementing the test

4.Automated unit testing tools facilitate these requirements

Boundary Value Analysis (BVA)

- Boundary value analysis is based on checking between partitions at the boundaries.
- This includes maximum, minimum, limits inside or outside, typical values, and error values.
- A large number of errors are generally seen to occur at the boundaries of the given input values rather than at the middle.
- It is also referred to as BVA and offers a number of test cases that exercise bounding values.

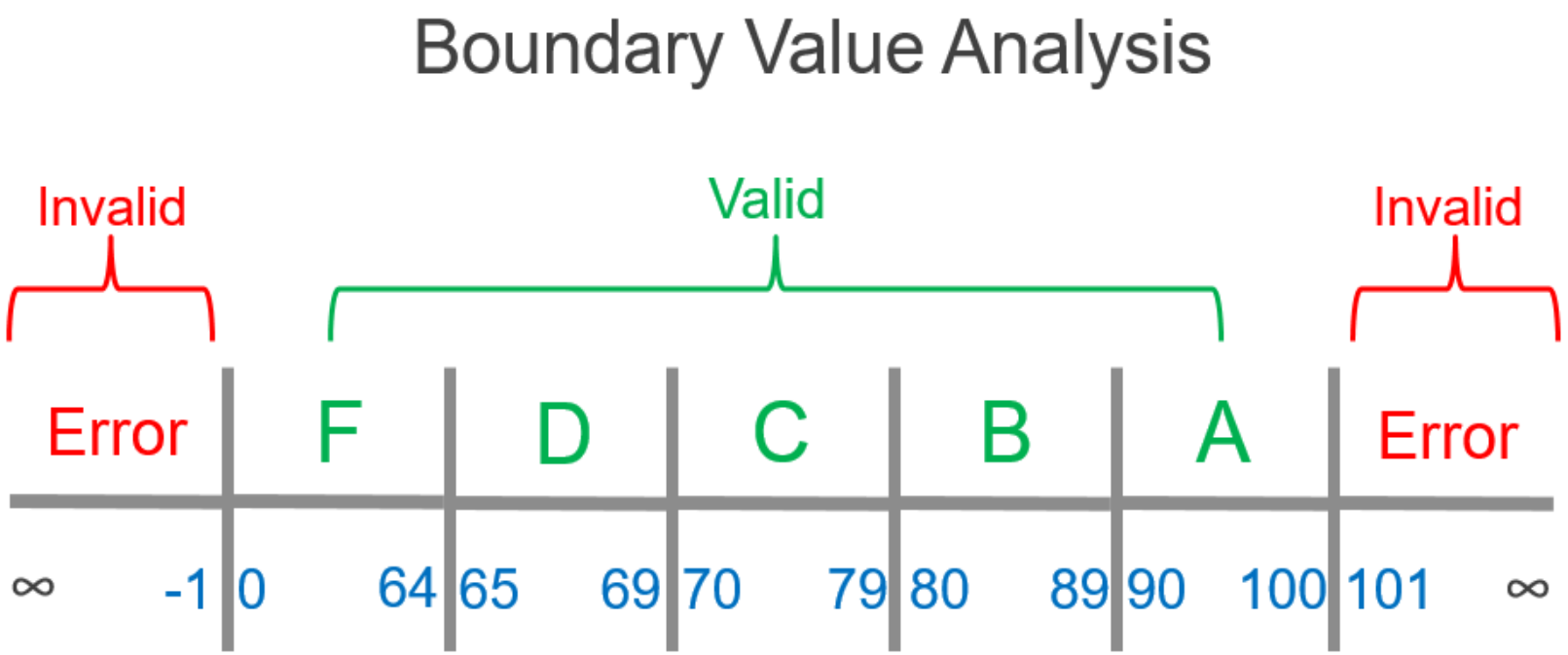
Boundary Value Analysis (BVA)

•This black box testing technique complements equivalence partitioning..

•This software testing technique base on the principle that, if a system works well for these particular

•values then it will work perfectly well for all values which comes between the two boundary values.

Boundary Value Analysis (BVA)



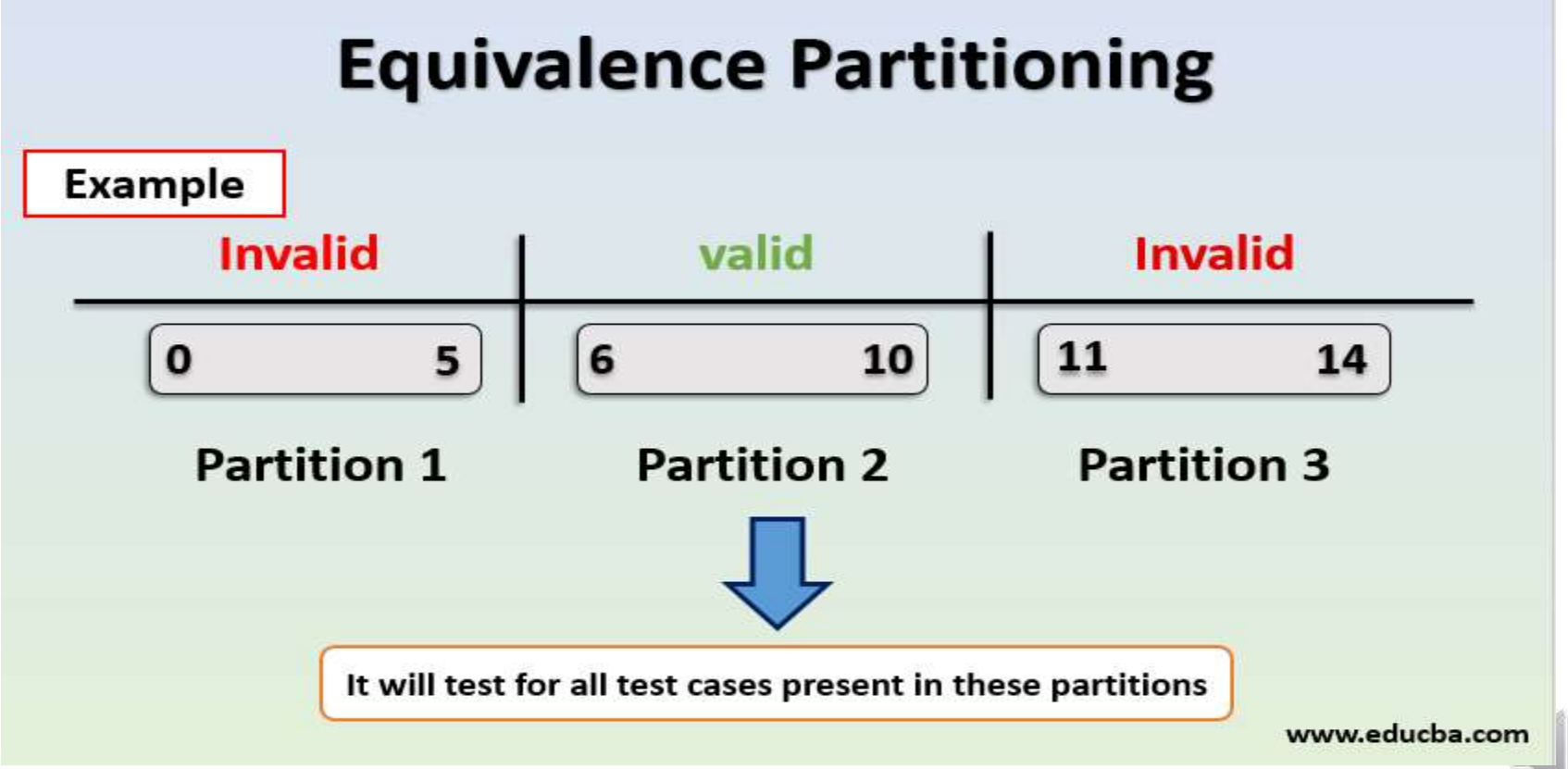
Guidelines for Boundary Value Analysis (BVA)

- If an input condition is limited to x and y values, then the test cases should be constructed with x and y values, as well as values above and below x and y values.
- If the input condition is a large number of values, the test case that the minimum and maximum numbers must be exercised should be established.
- Values above and below the minimum and maximum values are tested here as well.
- For performance conditions, apply guidelines 1 and 2. This produces an output that represents the minimum and the highest.

Equivalence Class Partitioning

- Equivalent Class Partitioning enables you to break the test condition set into a partition that should be treated as the same.
- This approach for software testing divides a programme's input domain into groups of data from which test cases should be designed.
- The idea behind this strategy is that the test case is equal to a test with some other value of the same class with a representative value
- It is of each class that a value will be used.
- It allows both true and invalid equivalence groups to be defined.
- Example: Input conditions are valid between
  1 to 10 & 20 to 30

Equivalence Class Partitioning



Decision Table Based Testing.

- The Cause-Effect map is also known as a decision table.
- For functions which respond to a combination of inputs or events, this software testing technique is used.
- For example, the submit button should be activated if all the appropriate fields have been entered by the user.
- The first task is to define functionalities where a combination of inputs depends on the output.
- If there are broad combinations of inputs, then split them into smaller subsets that are helpful for handling a table of decisions.
- You need to construct a table for every function and list all types of input combinations and their respective outputs.
- This helps to describe a condition that the tester overlooks.

Decision Table Based Testing.

Decision Table - Example

Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Heck the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

Issues in Object Oriented Testing

• As they include OO principles, conventional research approaches are not explicitly applicable to OO programmes.

•This includes encapsulation, polymorphism, and inheritance.

•Basic unit of monitoring for units.

•For unit test case design, the class is the natural unit.

•Apart from their class, the strategies are useless.

•A class in isolation can be tested by evaluating a class instance (an object).

•When independently validated classes are used in an application framework, they generate more complex classes.

Issues in Object Oriented Testing

• Before it can be deemed to be checked, the entire subsystem must be evaluated as a whole.

•2. Encapsulation Implication.

•Encapsulation of class attributes and methods can build barriers during testing.

•Since methods are invoked by the object of the corresponding class, it is not possible to perform testing without an object.

•The state of the object influences its actions at the time of the method's invocation.

•Testing is not only dependent on the object, but also on the state of the object, which is very complicated.

Issues in Object Oriented Testing

3Inheritance Implication.

Inheritance presents issues not present in conventional software.

The derived class does not always refer to test cases built for the base class.

In order to work properly in an OO environment, most testing methods need some kind of adaptation.

4. Polymorphism Consequences

A separate collection of test cases is needed for each possible binding of the polymorphic variable.

Before a client class can be checked, several server classes can need to be i

Issues in Object Oriented Testing

- Determining such bindings is difficult.
- It complicates preparation and testing for integration.

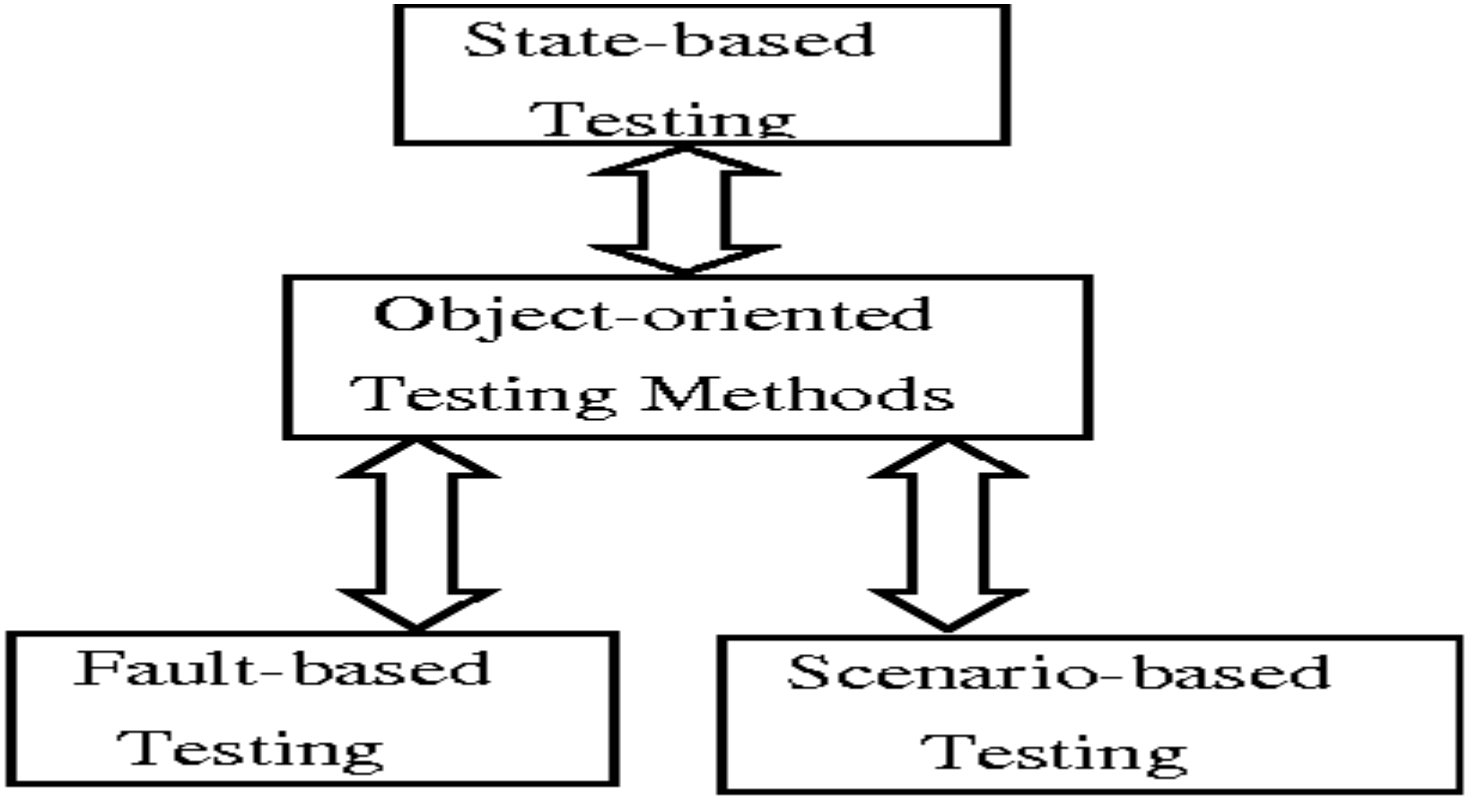
- 5. Implications for processes of research

- Both research methods and procedures need to be re-examined.

Fault Based Testing

- In an OO method, the purpose of fault-based testing is to design tests that have a high probability of uncovering probable faults.
- The product or device must adhere to the requirements of the consumer.
- With the research model, the preliminary preparation needed to conduct fault-based testing starts.
- Plausible faults (i.e. aspects of system implementation that may result in defects) are looked for by the tester.
- Test cases are designed to exercise the practise of these faults to assess if these faults exist design or code.

Fault Based Testing



Fault Based Testing

- We design test cases for each plausible fault that will cause the incorrect expression to fail.
- The efficacy of these strategies depends on how a "plausible defect" is viewed by testers.
- If real faults are considered to be "unplausible" in an OO system, then this method is really no better than any random testing technique.
- If design and analysis may provide insight into what is likely to go wrong.
- With relatively low effort expenditures, it may find large numbers of errors.

Fault Based Testing

.

•An object's "behaviours" are specified by the values assigned by its attributes.

•To decide whether proper values exist for different types of object actions, testing can use the attributes

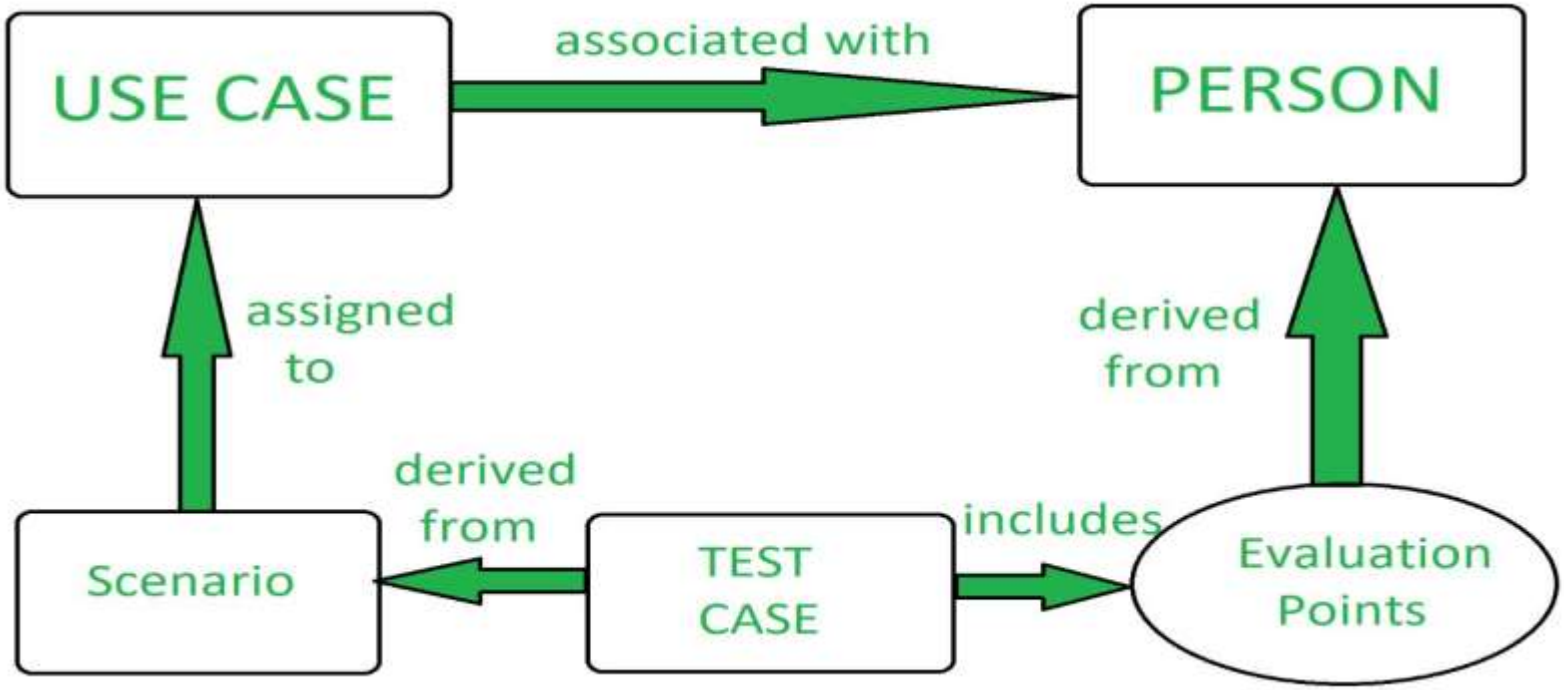
•It is important to remember that checking for integration seeks to identify errors in the client object, not the server.

•Integration testing focuses on deciding whether there are bugs in the calling code, not the code that is called.

Scenario Based Testing

- Two key forms of errors are ignored by fault-based testing
- (i)incorrect criteria and conditions
- (ii)The interactions between subsystems
- The product does not do what the consumer needs when mistakes associated with incorrect requirements occur.
- It could do the wrong thing or it could omit substantial features.
- But in either case, consistency (compliance with requirements) suffers from
- Errors associated with subsystem interaction arise when one subsystem 's activity happens.
- This generates situations(events, data flow) that cause another subsystem to fail

Scenario Based Testing



Scenario Based Testing

- Scenario-based research focuses on what the consumer is doing, not what the object is doing.
- Scenarios uncover mistakes in conversation.
- Yet test cases must be more complicated and more practical than fault-based experiments to achieve this.
- In a single test, scenario-based training aims to exercise different subsystems.
- Think of the creation of scenario-based tests for a text editor as an example. Usage cases follow cases
- Use-Case: Print a copy of a new one

Scenario Based Testing

Background: Someone asks for a new copy of the document from the user.

1. Open this document.
2. Just print it.
3. Close down a text.

The approach to research is reasonably clear.

Except that it didn't come out of nowhere in this text.

In an earlier mission, it was generated. Will this assignment impact this one?

But this example implies a possible flaw in the specification.

The editor does not do what the consumer wants it to do fairly.

The review noted in phase 3 above is frequently missed by customers.

Scenario Based Testing

- When they trot off to the printer and find one page when they want 100, they will then be irritated.
- Annoyed customers report bugs in specifications
- This reliance in test design may be ignored by a test case designer
- But during testing, it is possible that the issue will emerge.
- Then the tester will have to deal with the likely answer, "That's the way it's supposed to work,"

Random Testing

- A type of functional black box testing is Random Testing, also referred to as monkey testing.
- That is achieved when there is not enough time for the tests to be written and executed.
- It is assisted by the development of a random test sequence that attempts the minimum number of operations typical of the categories' actions.
- Find a banking application to include brief examples of these approaches.
- In which the following operations are performed by an account class: open, setup, deposit, withdraw, balance, overview.

Random Testing

•For example, the account must be opened before it is possible to apply and close other operations after all operations have been completed.
•Even with these restrictions, the operations have several permutations.
•The minimum history of behavioural life of an account example contains the following operations:
•Open • configuration • deposit • withdraw •close
•This reflects the minimum account checking sequence.
•Within this series, however, a wide variety of other behaviours may occur:
•Open Setup Deposit Deposit • Open Setup Deposit.

Random Testing

You can randomly produce a number of different procedure sequences. For instance:
Test case r1: open • configuration • deposit • deposit • balance • summarise • withdraw • close • close
Test case r2: open • configuration • deposit • withdraw • deposit • balance • credit Cap • withdraw • close

This and other random order experiments are carried out to carry out distinct life histories of class instances.

Partition Testing

- Partition training decreases the number of test cases that are appropriate for the class to exercise.
- In much the same way as traditional device equivalency partitioning.
- Input and output are classified and test cases are planned for each group to be exercised.
- State-based partitioning classifies class operations on the basis of their ability to modify the class state.
- Again, state operations require deposit and withdrawal, given the account class.
- Whereas non-state behaviours include equilibrium, summarize, and credit Limit.

Partition Testing

- Tests are structured in a way that conducts state-changing operations and those that do not separately alter the state. Accordingly,
- Test case p1: open • setup • deposit • deposit • withdraw • withdraw • close Configuration • deposit • withdraw
- Test case p2: open • configuration • deposit • summarise • credit cap • withdraw • close • close
- Test case p1 alters status, while test case p2 conducts operations that do not modify status (other than those in the minimum test sequence).
- For each partition, test sequences are then built.
- Category-based partitioning categorizes class operations based on the generic function that each performs.

Interclass Test Case Design

- As integration of the OO system starts, test case design becomes more complicated.
- It is at this point that testing must begin for partnerships between classes.
- We extend the banking example introduced in Section 23.5 to include the classes in order to demonstrate 'interclass test case generation.'
- The orientation of the arrows in the figure shows the direction of the messages, and the operations are shown by the dot.
- As a result of the partnerships implied by the messages, that are invoked.

Interclass Test Case Design

- Like the testing of individual students, it is possible to conduct class collaboration testing by applying randomly.
- Methods of partitioning, as well as scenario-based research and testing of actions.

Refrences

[1]. Roger Pressman, Software engineering- A practitioner's Approach, McGraw-Hill International Editions
[2]. Yogesh Singh, Software Testing, Cambridge University Press.
[3]. William E. Perry, Effective Methods for Software Testing, Second Edition, John Wiley & Sons
[4]. Ron Paton, Software Testing, second edition, Pearson education.
[5] Dohorthy Graham foundation of Software Testing,ISTQB Certification
[6] Softwaretesting. Tutorialspoint.
https://www.tutorialspoint.com/Software testing

Parul® University

NAAC GRADE A++

https://paruluniversity.ac.in/