EXPERIMENT - 3

Software Testing and Quality Assurance

Aim

Equivalence Class Partitioning In the lecture, we used the example of an app that classified Risk Exposure (RE) as High, Moderate, or Low on the basis of Risk Probability (RP) and Risk Impact (RI).

Consider the following specification for such an app:

1) the app accepts two integers, RP and RI, as input,
2) both RP and RI must be in the range [1, 5],
3) if either input is not an integer, it is invalid and the app outputs "Invalid,
4) if either input is an integer outside the range specified in (2), it is invalid and the app outputs "Out of Range",
5) given valid inputs, the app calculates RE as the product of RP and RI, and outputs:
   a. "High", if RE is greater than 9
   b. "Low" if RE is less than or equal to 2
   c. "Moderate" if neither (a) nor (b) are satisfied

.

Syeda Reeha Quasar
14114802719

EXPERIMENT – 3

Aim:

Equivalence Class Partitioning In the lecture, we used the example of an app that classified Risk Exposure (RE) as High, Moderate, or Low on the basis of Risk Probability (RP) and Risk Impact (RI).

Consider the following specification for such an app:

1) the app accepts two integers, RP and RI, as input,

2) both RP and RI must be in the range [1, 5],

3) if either input is not an integer, it is invalid and the app outputs "Invalid,

4) if either input is an integer outside the range specified in (2), it is invalid and the app outputs "Out of Range",

5) given valid inputs, the app calculates RE as the product of RP and RI, and outputs:

   a. "High", if RE is greater than 9

   b. "Low" if RE is less than or equal to 2

   c. "Moderate" if neither (a) nor (b) are satisfied

i) Partition the domain of each parameter into equivalence classes, label the classes, and list them.

ii) Develop a set of test cases for the app to satisfy Each Choice Coverage of the equivalence classes. Indicate the equivalence classes covered by each test case and, as always, include the expected result. Notice the actual classification process is not adequately tested by your set of test cases.

iii) To better test the classification performed by the app, partition the output domain and develop additional test cases to cover any class not covered by your test cases in (ii).

Theory:

Risk is the probability of a negative or undesirable outcome or event. Risk is any problem that might occur that would decrease customer, user, stakeholder perception of quality and/or project success.

Types of risks:

There are 2 types of risks- Product Risk and Quality Risk.

1. Product risk-
   When the Primary effect of a potential problem is on product quality, then the potential problem is called Product risk. It can also be called a quality risk. Example- A defect that can cause a system crash or monetary loss.

2. Project Risk-
   When the Primary effect of a potential problem is on the overall success of the project, those potential problems are called Project risk. They can also be called Planning risks. Example- Staffing issues that can delay project completion.

Not all risks are equally important. There is a number of ways we can classify the level of risk. The easiest way is to look at two factors-

1. Likelihood of occurrence of the problem. Likelihood arises from technical considerations.
2. Impact of the problem, if it occurs. Impact arises from business considerations.

What Is Risk-Based Testing?

In risk-based testing, we use the risk items identified during risk analysis, along with the level of risk associated with each risk item to guide the testing. It is a type of software testing technique that is primarily based on the probability of the risk. Risk-based testing involves the following steps:

1. Accessing the risk based on software quality.
2. Frequency of use.
3. Criticality of Business.
4. Possible areas with defects, etc.

Characteristics Of Risk-Based Testing (RBT):

Below are some characteristics of Risk-based testing (RBT)-

1. RBT strategy matches the level of test effort to the level of risk. The higher the risk, the more is the test effort. This applies to test execution as well as other test activities like test designing and implementation.
2. It matches the order of testing with the level of risk. Higher risk tests tend to find more defects or are related to more important areas in the application or maybe both. So higher the risk, we plan the tests earlier in the cycle- both in design and execution. This helps in building the confidence of business stakeholders as well.
3. By effort allocation and maintaining the order of testing, the quality risk gets systematically and predictably reduced. By maintaining a traceability matrix of tests vs risks and defects identified to risks, reporting the risk as residual risks make sense. This allows the concerned stakeholders to decide when to stop testing i.e whenever the risk of continuing testing exceeds the risk of testing completion.

4. If schedule reduction requires scope reduction, it is easier to decide what needs to go out. It will always be acceptable and explainable to business stakeholders as risk levels are agreed upon.
5. To identify risks, we need to take inputs from various sources like – Requirements specifications, design specifications, existing application data, previous incident records, etc. However, if this information is not readily available, we can still use this approach by getting inputs from stakeholders for the risk identification and assessment process.

When To Implement Risk-Based Testing?

Risk-based testing approach is implemented in scenarios where-

1. There is time/resource or budget constraints. For example- A hotfix to be deployed in production.
2. When a proof of concept is being implemented.
3. When the team is new to the technology platform or to the application under test.
4. When testing in Incremental models or Iterative models.
5. Security testing is done in Cloud computing environments.

How Risk-Based Testing Is Implemented?

Risk can guide testing in multiple ways but below are the major ones –

1. The effort is allocated by test managers proportional to the risk associated with the test item.
2. Test techniques are selected in a way that matches the rigor and extensiveness required based on the level of risk of the test item.
3. Test activities should be carried out in reverse order of risk i.e The Highest risk item should be tested first.
4. Prioritization and resolution of defects should be done as appropriate with the level of risk associated.
5. During test planning and control, test managers should carry out *risk control* for all significant risk items. The higher the level of risk associated, the more thoroughly it should be controlled.
6. Reporting should be done in terms of residual risks. Example- Which tests have not run yet? Which defects have not fixed yet?

Benefits of Risk-Based Testing (RBT):
By identifying and analyzing the risks related to the system, it is possible to make testing efficient and effective-

1. Efficient-
   RBT is efficient because you test the most critical areas of the system early in the cycle (the earlier the defect is detected the lower is the cost of solving those issues)
2. Effective-
   RBT is effective because your time is spent according to the risk rating and mitigation plans. You do not spend time on items and activities which might not be very important in the overall scheme of things.
3. Reduced Test cases-
   Test case count gets reduced as test cases become more focused.
4. Cost reduction-
   A reduction in cost per quality as critical issues get fixed early in the cycle and hence lowering the cost of change.
5. Faster time to market-
   Faster time to market is more easily achievable with RBT approach as the most important features are available in a shippable position early in the cycle.

Solution:

i)

RP	RI	Equivalence Class
RP < 1	1 <= RI <=5	Invalid
RP > 5	1 <= RI <=5	Invalid
1 <= RP <=5	RI < 1	Invalid
1 <= RP <=5	RI > 5	Invalid
Non Integer	1 <= RI <=5	Invalid
Non Integer	RI < 1	Invalid
Non Integer	RI > 5	Invalid

RP < 1	Non Integer	Invalid
RP > 5	Non Integer	Invalid
1 <= RP <=5	Non Integer	Invalid
1 <= RP <=5	1 <= RI <=5	Valid

ii)

RP	RI	Equivalence Class	Test Case RP	Test Case RI	Expected Output
RP < 1	1 <= RI <=5	Invalid	0	3	Out of Range
RP > 5	1 <= RI <=5	Invalid	7	2	Out of Range
1 <= RP <=5	RI < 1	Invalid	3	-1	Out of Range
1 <= RP <=5	RI > 5	Invalid	4	23	Out of Range
Non Integer	1 <= RI <=5	Invalid	23.4	3	Invalid
Non Integer	RI < 1	Invalid	a	0	Invalid
Non Integer	RI > 5	Invalid	Hello	17	Invalid
RP < 1	Non Integer	Invalid	-45	3.2	Invalid
RP > 5	Non Integer	Invalid	60	xyz	Invalid
1 <= RP <=5	Non Integer	Invalid	2	*	Invalid
1 <= RP <=5	1 <= RI <=5	Valid	2	3	Moderate

iii)

Test Case RP	Test Case RI	Expected Output
4	3	High
1	1	Low
3	2	Moderate

Result:
Problem based on the given conditions was solved and verified.

Viva Questions

Q1. What is risk-based testing?.
Risk-based Testing is the term used for an approach to creating a Test Strategy that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

Q2. How to perform risk based testing?

1. Make a prioritized list of risks.
2. Perform testing that explores each risk.
3. As risks evaporate and new ones emerge, adjust your test effort to stay focused on the current crop.

Q3. How to improve skills designing test cases and make sure high coverage rate?

Test designing is successful when the requirements are analyzed and understood completely. To ensure 100% test coverage is achieved, you should

not miss out on creating test cases for any requirements and from time to time we can check ourselves with the help of a traceability matrix.

Q4. What's the difference between System testing and Acceptance testing?

Acceptance testing checks the system against the "Requirements." It is similar to System testing in that the whole system is checked but the important difference is the change in focus:

System testing checks that the system that was specified has been delivered.

Acceptance testing checks that the system will deliver what was requested. The customer should always do Acceptance testing and not the developer.

The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgement. This testing is more about ensuring that the software is delivered as defined by the customer. It's like getting a green light from the customer that the software meets expectations and is ready to be used.

Q5: How do you define a testing policy?

The first step any organization needs to do is define one unique definition for testing within the organization so that everyone is of the same mindset.