Modeling A TEST Design Process

MODELING A TEST DESIGN PROCESS

Create State It is the creator of a test case, also referred to as the owner or creator, who is responsible for initiating the creation of the test case and putting it into this original condition. The following required variables affiliated with the test case are initialized by the creator: requirement_ids, tc_id, tc_title, originator_group, creator, and test_category. It is anticipated that the test case will validate the requirements that are listed in the field labeled "requirement_ids." The group that first recognized the need for the exam is known as the originator group. By filling out the eng_assigned column, the creator of the test case has the ability to allocate it to a particular test engineer, including himself, and to transition the test case from the create state to the draft state.

Draft State The test group, also known as the system test squad, is the entity in charge of maintaining this condition. During this phase of the process, the designated test engineer is responsible for entering the following data: tc_author, objective, setup, test_steps, cleaning, pf_criteria, candidate_for_automation, and automation_priority. After the creator of the test case has finished filling out all of the required sections, the test engineer may transfer the test case to the creator so that the creator can run through the test case. The test case will remain in this condition until the developer examines it and moves through each step. After that, the creator has the ability to change the status of the document from the draft state to the review state by entering all of the names of the approvers into the section labeled "approver_names."

States That Were Reviewed and Eliminated The person who initially developed the test scenario is also the proprietor of the evaluation state. The proprietor extends an invitation to the test engineers and developers to look over the test case and

validate it. They check to see that the test case can be run, and that the conditions for passing or failing are articulated in detail. In the event that any of the fields require an update, action items pertaining to the test scenario are formulated.
Action items resulting from a review conference are entered into the review_actions section, and the owner of the test case is responsible for carrying out the action items in order to implement modifications. After receiving positive feedback from all of the evaluators, the test case is moved into the released stage. In the event that the evaluators come to the conclusion that this is not a legitimate test case or that it cannot be executed, the test case will be shifted to the deleted state. In order to remove a test case, there must first be a review action item that states "delete this test case."

States That Are Both Released and Up to Date When a test case is in the published state, it is in a state where it is ready to be run, and it is added to a test suite. On the other hand, a test case that is in the update state indicates that it is currently undergoing one or more of the following processes: having its reusability improved; having its pass–fail parameters fine-tuned; and/or having the specific test procedure fixed. For instance, rather than hard coding the data values, a repeatable test case ought to make use of parameters instead. In addition, a test case needs to be changed so that it can be adapted to changes in the surroundings or the functionality of the system. By taking a test case through the released– update cycle a limited number of times, one can improve the reproducibility of the test case, thereby making it easier for other people to rapidly comprehend, borrow, and reuse it. In addition to this, this lays the groundwork and justifies the need for the test case to be mechanized. It is important for a test scenario to be platform agnostic. After implementing a remedy for a bug, a test engineer might return a test case to its previously released state if an upgrade only introduces a minor

alteration. In the event that this is not the case, the test case will be put through an additional review, which can be accomplished by shifting it to the review stage.
Every time a test case is run, it is possible to make one revision to it.

Former State Now Deprecated A test case that is no longer relevant could be shifted to a deprecated state. In an ideal scenario, the test case should be examined to determine whether or not it continues to exist if it has not been run for at least a year. Because of the factors listed below, a test case might become irrelevant over the course of time. First, the functionality of the system being tested has undergone significant revisions, and second, an insufficient amount of test case maintenance has led to an outdated test case. Second, when an older test case is changed, it's possible that some of the requirements of the first test case will no longer be met.
Third, as the circumstances evolve over time, the reusability of test instances has a tendency to deteriorate. This is particularly true of test cases that were not developed with sufficient consideration given to the possibility of reuse. In conclusion, test cases may be haphazardly brought forward for a considerable amount of time after the original justifications for them have vanished. It's possible that nobody knows the reason a particular test case was created in the first place, but that doesn't stop people from using it.

After a test factory has been established, a test manager is able to develop a test suite with the assistance of a test suite template. In order to combine test cases for the purpose of evaluating a specific version, a test suite schema is utilized, as can be seen in Table 11.7. A test suite ID, a title, an objective, and a collection of test cases that need to be handled by the test suite are all necessary components of the structure. Additionally, one determines the individual test cases that will need to be carried out (test rounds 1, 2, 3, and/or regression), as well as the specifications that each test case must meet. The plan is to compile a new project's test suite from a subset of previously published test cases by first selecting a certain number of those cases and then repackaging them.

MODELING TEST RESULTS

After a test case has been designed or selected, its execution status is reset to its original state, which is unverified. This happens automatically. The test case outcome will be moved into the invalid state if the test case is not legitimate for thesoftware version that is currently being tested. In the condition that has not been tested, the identifier of the test suite is recorded in a variable that is called test_suite_id. Once the execution of a test case has begun, the status of the test outcome may transition into one of the following states: succeeded, failed, invalid, or delayed. If the test case processing is finished and it meets the pass criteria, a test engineer may transfer the test case outcome from the untested state to the passed state. This occurs when the test case is considered to have satisfied the passcriteria.

A test engineer will move the test result from the untested state to the failed state ifthe test execution is finished and has fulfilled the fail criteria. They will then correlate the defect with the test case by initializing the defect_ids field and transfer the test result to the failed state. When a new build is obtained that contains a remedy for the bug, the test case needs to be rerun so that the change can be verified. In the event that the reexecution is finished and the conditions for passing the test are met, the test outcome will be changed to the passed state.

If it is not feasible to completely carry out the test case's instructions, the outcome of the test is set to the stopped state. The defect number that prevents the test case from being executed is, if it is known, entered into the defect_ids section of the database. When a new build that addresses a stalled test case is obtained, the test case could potentially be reexecuted again. The test result is considered to have passed when it is shifted into the passed state provided that the processing has been finished and the conditions for passing have been met. If, on the other hand, it meets the requirements for failing the test, the outcome of the test will be shifted tothe failed state.

If the execution does not succeed because of a new stopping defect, the test resultwill continue to be in the stopped state, and the defect_ids field will be updated toinclude the new defect that prevented the test case from being run.

The test case preparation work that is being done by a team of test engineers may be of interest to the management in terms of its progression, level of coverage, andoverall effectiveness.

This information enables them to (i) determine whether or not a test project is progressing in accordance with the timetable and whether or not additional resources are necessary, and (ii) plan their subsequent project with greater precision. The degree of preparedness that can be represented by the test designcan be measured using the following measures.

The following outlines the Preparation Status of Test Cases (PST): Before being made available for use as a legitimate and helpful test case, a test case might be putthrough a number of stages, also known as states, such as "draft" and "review." As a result, it is beneficial to conduct frequent checks on the development of the test design by tallying the number of test cases that are located in various stages of design, including create, draft, review, released, and deleted. Before the beginning of test execution, it is customary to anticipate that all of the scheduled test cases that have been developed for a particular project will ultimately transition to the released state.

Average Time Spent (ATS) in Test Case Design: It is helpful to know the amount of time it takes for a test case to progress from its original conception, which is the

create state, to when it is considered to be useable, which is the released state. Thisinformation is referred to as the average time spent in the design of a test case. This measure is helpful in allocating time to the test preparation activity in a succeedingtest project, which is why it is being discussed here. As a result, it is beneficial to the preparation of the exam.

Number of Available Test (NAT) Cases: This refers to the total number of test scenarios that have been completed and published by previously completed initiatives. For the purpose of the ongoing testing endeavor, we have chosen to perform regression testing on some of these test scenarios.

The number of test cases that are part of a test package and are prepared for implementation before the beginning of system testing is referred to as the numberof planned test cases, or NPT cases.

This measure can be helpful when planning the implementation of tests. As the testing progresses, new test cases that weren't originally intended might need to be developed. When compared to NPT, the significant increase in the number of newtest instances suggests that the original planning was inaccurate.

A Test Suite's (CTS) Level of Coverage: This measure calculates the proportion of all requirements that are validated by the full set of test cases or by a specific number of those instances.

The critical test suite, or CTS, is a measurement of the number of test cases that need to be chosen or developed in order to have adequate coverage of system specifications.

The objectives of the test case design effectiveness metric are to (i) quantify the "defect revealing ability" of the test suite and (ii) use the metric to enhance the test design process. Both of these objectives are intended to be accomplished by using the metric. Whenever scheduled test cases are carried out, it is possible for defects to become apparent during system-level testing. During testing, in addition to thesedefects, new defects are discovered that had not been prepared for by any of the test scenarios. These newly discovered flaws require the development of brand new test cases, which are referred to as test case escaped. (TCE). Test failures are usually the result of flaws in the procedure used to create the tests. This occurs as aresult of the test engineers coming up with novel concepts while they are carrying out the scheduled test cases. The test case design yield, abbreviated as TCDY, is a metric that is frequently used in the business to measure the effectiveness of test case design.