UNIT - IV

Software Testing

Topics

- Introduction to Testing
- Verification
- Validation
- Test Strategy
- Planning
- Example – Test Strategy and Planning
- Test Project Monitoring and Control
- Design – Master Test Plan, Types
- Test Case Management
- Test Case Reporting

Software Testing

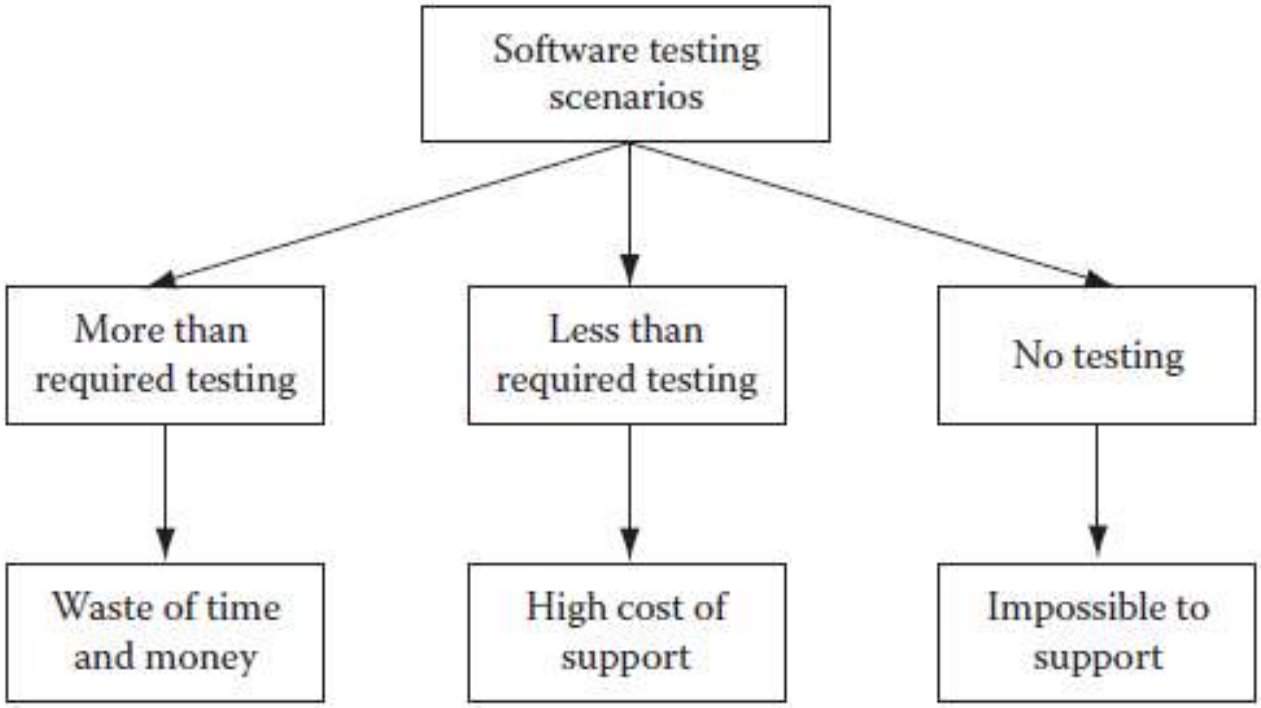
Introduction

- It is a fact that the exact number of defects in a software product is difficult to find.
- At best it can be predicted using some defect estimation tools.
- It is also impossible to detect all defects in a software product.
- Nevertheless, finding and fixing critical bugs up to an acceptable limit as per expectations is important.
- If there are more defects in the product after the product enters production, then the project team will be in big trouble.
- The support costs for a bug ridden product will be too high.
- So, less than required testing is a certain call for rebuke from stakeholders.
- Testing more than required will increase project costs unnecessarily.
- When the project starts, the customer specifies what level of quality for the product is expected.
- The project manager needs to first make sure that the processes to be followed for building the product are at least so good that the produced product will have a certain level of quality with a certain level of defects.

Software Testing

Introduction

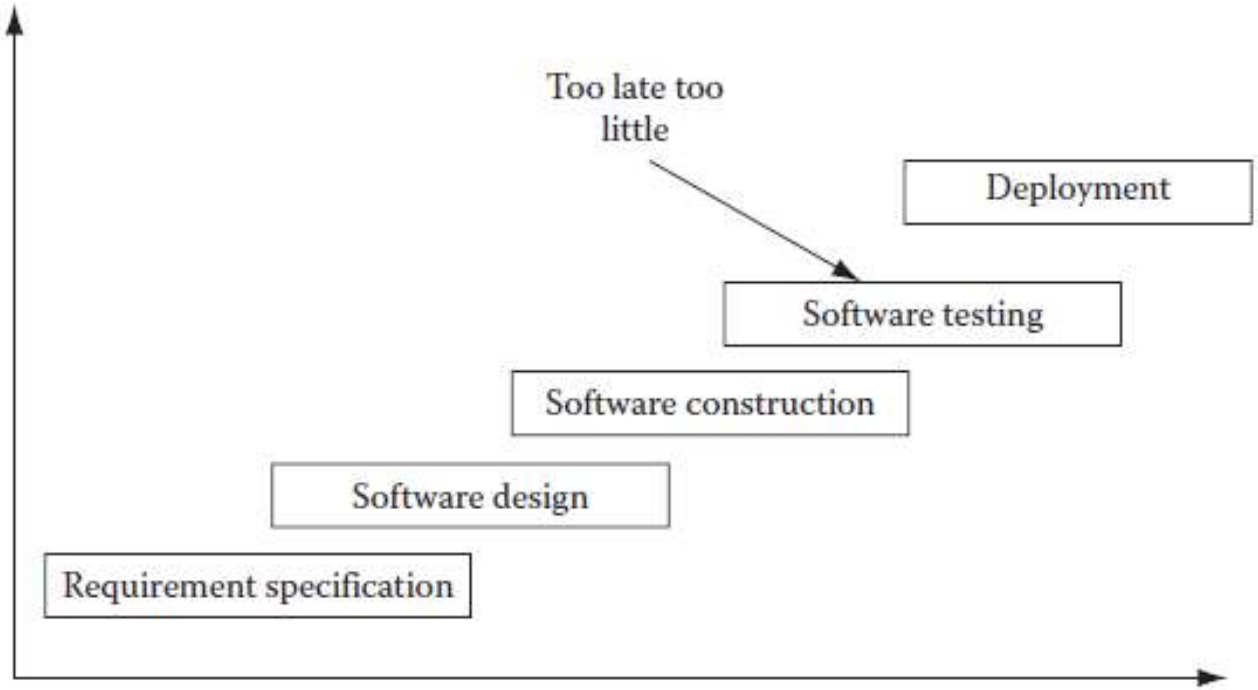
- Then, he should have a test plan such that the product defects are further reduced by finding defects and fixing them.
- So the testing phase must be well planned with required budget, schedule and testing processes that will ensure that a certain number of critical defects are caught and fixed (Figure below-Software Testing Scenarios).



Software Testing

Introduction – Problems with Traditional Development Model

- Traditionally, software testing was done only after software was constructed.
- This is used to limit the scope of software testing in the development life cycle (see Figure below-Traditional Software Development Model-Too little, Too late testing).
- This practice led to a situation that was too little and too late.
- By the time software was constructed, already faulty requirement specifications and faulty software design had resulted in defect ridden software.



Software Testing

Introduction – Problems with Traditional Development Model

- Removing all the defects originating from different phases of the project in one go is a huge challenge.
- That is why this approach always used to result in defect ridden software products.
- Even if there was an attempt to remove defects so late in the life cycle, it would be exorbitantly costly to do so in one go and it would also mean devoting a considerable amount of time in detecting and fixing all those defects.
- This would likely be infeasible.
- Definitely a better approach was needed to make better quality software products.

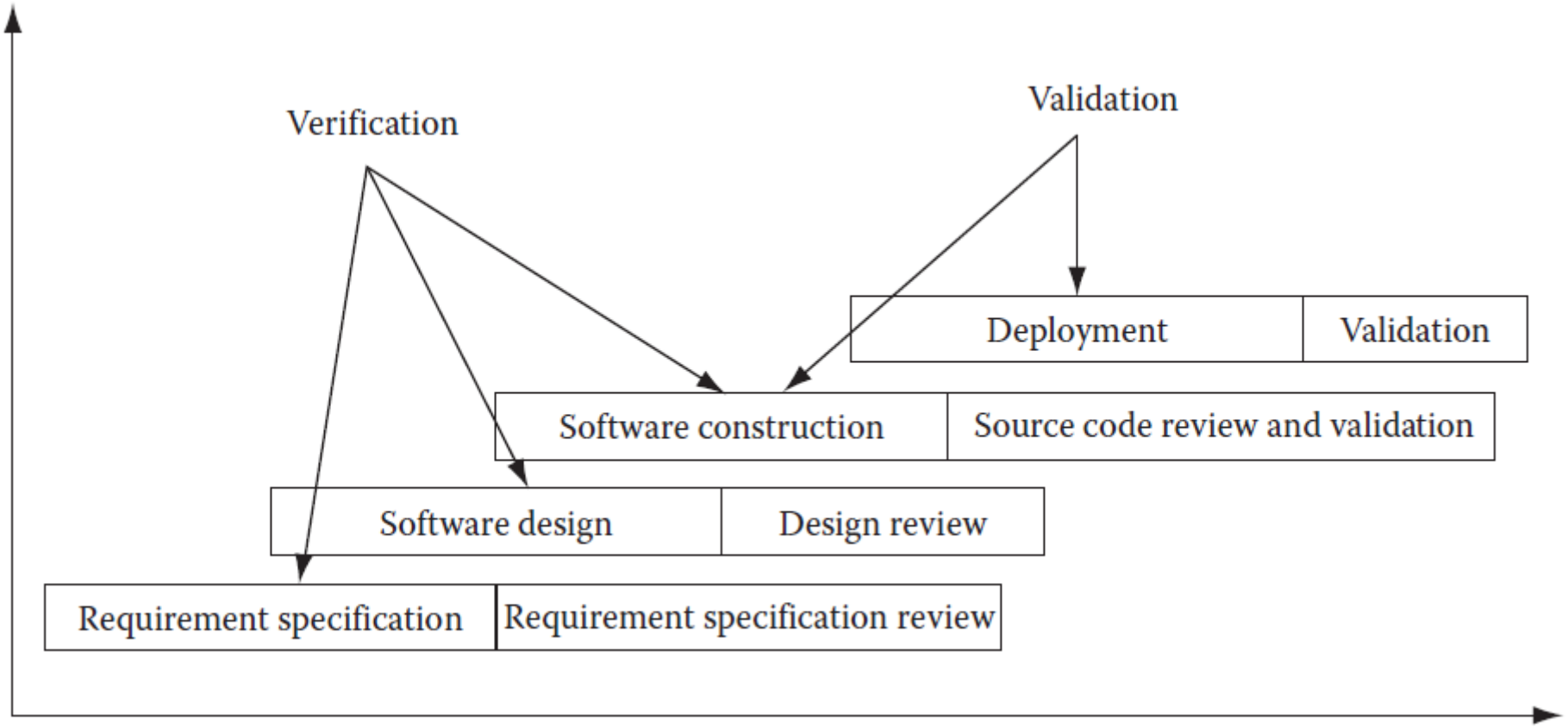
Software Testing

Verification

- The problems encountered in the traditional approach to software testing led to the practice of verification and validation.
- In most quality standards documents, software testing is divided into two parts: “validation” and “verification.”
- While verification implies that the developed software is working as intended by checking the requirement specifications, design, source code, etc., in static mode, validation implies that the software has been validated to be working after running it and checking whether all functionality meets the requirements.
- Verification techniques are also known as static testing, since the source code is not run to do testing.
- The figure below (Software verification and validation) shows that each work product including requirement specifications, design, and source code during software development is tested using static methods.
- The requirement specifications are reviewed for completeness, clarity, design ability, testability, etc.
- The software design is reviewed for robustness, security, implementability, scalability, complexity, etc.

Software Testing

Verification and Validation



- The source code is reviewed for dead code, unused variables, faulty logic, constructs, etc.
- Once the source code is ready to be run as a system, validation testing can be started.

Software Testing

Validation

- Validation testing is also known as dynamic testing as, in this case, the source code is actually run to determine that it is running per specifications.
- During validation, unit, integration, system, and finally user acceptance testing are performed.
- Unit testing is done to ensure each unit piece of source code is free from defects.
- Once unit testing is done, then this piece of code is integrated with the main source code build.
- But before integrating to the main build, it is strongly advisable to do local integration testing on the developer's own computer.
- Only when the source code runs smoothly and all integration tests pass, the source code should be integrated with the main build.
- When all source code is thus integrated, the main build is ready for system testing.
- All system tests are then performed and defects are fixed.
- When the system testing is over and in fact the software product is shipped to customers, they do user acceptance testing.

Difference Between Verification and Validation

Verification	Validation
Evaluates the intermediary products to check whether it meets the specific requirements of the particular phase.	Evaluates the final product to check whether it meets the business needs.
Checks whether the product is built as per the specified requirement and design specification.	It determines whether the software is fit for use and satisfies the business needs.
Checks “Are we building the product right”?	Checks “Are we building the right product”?
This is done without executing the software.	Is done with executing the software.
Involves all the static testing techniques.	Includes all the dynamic testing techniques.
Examples include reviews, inspection, and walkthrough.	Example includes all types of testing like smoke, regression, functional, systems and UAT.

Software Testing

Test Strategy and Planning

- Software testing is a vast field in itself, and so the common practice is to consider it as a separate project.
- In those cases, it is known as an independent verification and validation project.
- As such, a separate project plan is made for that project and is linked to the parent software development project.
- There are many techniques available to execute software test projects.
- It depends on the kind of test project. However, most test projects must have a test plan and a test strategy before the project can be ready for execution.
- Often due to time constraints, testing cycles are cut short by project managers.
- This leads to a half-tested product that is pushed out of the door.
- In such cases, a large number of product defects are left undetected.
- Ultimately, end users discover these defects. Fixing these defects at this stage is costly.
- Moreover, they cannot be fixed one at a time.
- They are to be taken in batches and are incorporated in maintenance project plans.
- This leads to excessive costs in maintaining the software.

Software Testing

Test Strategy and Planning

- It is a lot cheaper to trap those bugs during the testing cycle and fix them.
- It is appropriately said that “testing costs money but not testing, costs more!”
- Test strategies should include things like test prioritization, automation strategy, risk analysis, etc.
- Test planning should include a work breakdown structure, requirement review, resource allocation, effort estimation, tools selection, setting up communication channels, etc.

<u>Test Prioritization</u>

- Even before the test effort actually starts, it is of utmost importance that the test prioritization should be made.
- First of all, all parts of the software product will not be used by end users with the same intensity.
- Some parts of the product are used by end users extensively, while other parts are seldom used.
- So the extensively used parts of the product should not have any defects at all and thus they need to be tested thoroughly.

Software Testing

Test Strategy and Planning - Test Prioritization

- For making such a strategy, you must prioritize your testing.
- Put a high priority on tests which are to be done for these critical parts of the software product and put a low priority on uncritical parts.
- Then test the high priority areas first.
- Once testing is thoroughly done for these parts, then you should start testing the low priority areas.

Test Strategy and Planning – Risk Management

- The test manager should also do plan for all known risks that could impact the test project.
- If proper risk mitigation planning is not done and a mishap occurs, then the test project schedule could be jeopardized, costs could escalate and/or quality could go down.
- Some of the risks that can have severe, adverse impact on a test project include an unrealistic schedule, resource unavailability, skill unavailability, frequent requirement changes, etc.

Software Testing

Test Strategy and Planning – Risk Management

- Requirement changes pose a serious threat to testing effort because for each requirement change, the whole test plan gets changed.
- The test team has to revise its schedule for additional work as well as to assess impact of the change on the test cases they have to recreate.
- Some enthusiastic test engineers estimate much less effort than it actually should be.
- In that case, the test manager would be in trouble trying to explain why testing is taking more than the scheduled time schedule.
- In such cases, even after loading testing engineers more than 150%, the testing cycle get delayed.
- This is a very common situation on most of the test projects.
- This also happens because the marketing team agrees on unrealistic schedules with the customer in order to bag the project.
- Even the test manager at that time feels that somehow he will manage it, but later on it proves impossible to achieve.
- Other test engineers unnecessarily pad their estimate and later on, when the customer detects it, the test manager finds himself in a spot.

Software Testing

Test Strategy and Planning – Risk Management

- When the software development market, along with the software testing market, is hot (this is the case most of the time, as businesses need to implement software systems more and more and so software professionals are in great demand), software professionals have many job offers in hand.
- They leave the project at short notice and the test manager has to find a replacement fast.
- Sometimes, a project may have some kind of testing for which skilled test professionals are hard to find.
- In both situations, the test manager may not be able to start those tasks in need of adequate resources.
- For test professional resources, a good alternative resource planning is required.
- The test manager should be in consultation with human resource manager, keep a line of test professionals who may join in case one is needed on his project.
- For scheduling problems, the test manager has to ensure in advance that schedules do not get affected.
- He has to keep a buffer in the schedule for any eventuality.

Software Testing

Test Strategy and Planning – Risk Management

- To keep a tab on the project budget, the test manager has to ensure that the schedule is not unrealistic and also has to load his test engineers appropriately.
- If some test engineers are not loaded adequately, then project costs may go higher.
- For this reason, if any test professionals do not have enough assignments on one project, they should be assigned work from other projects.

Test Strategy and Planning – Effort Estimation

- For making scheduling, resource planning and budget for a test project, the test manager should make a good effort estimate.
- Effort estimate should include information such as project size, productivity, and test strategy.
- While project size and test strategy information comes after consultation with the customer, the productivity figure comes from experience and knowledge of the team members of the project team.
- The wideband Delphi technique uses brainstorming sessions to arrive at effort estimate figures after discussing the project details with the project team.

Software Testing

Test Strategy and Planning – Effort Estimation

- This is a good technique because the people who will be assigned the project work will know their own productivity levels and can figure out the size of their assigned project tasks from their own experience.
- Initial estimates from each team member are then discussed with other team members in an open environment.
- Each person has his own estimate.
- These estimates are then unanimously condensed into final estimate figures for each project task.
- In an experience-based technique, instead of group sessions, the test manager meets each team member and asks him his estimate for the project work he has been assigned.
- This technique works best when team members are well aware, particularly, of their prior experience of similar project tasks.
- Effort estimation is one area where no test manager can have a good grasp at the initial stages of the project.
- This is because not many details are clear about the project.
- As the project unfolds, after executing some of its related tasks, things become clearer.

Software Testing

Test Strategy and Planning – Effort Estimation

- At that stage, any test manager can comfortably give an effort estimate for the remaining project tasks. But that is too late.
- Project stakeholders want to know at the very beginning of the project, what would be the cost estimates and when the project would be delivered.
- These two questions are very important for project stakeholders and it is on the top of their mind.
- Unfortunately, test managers are not equipped to provide an accurate schedule and costs for the project at those initial stages because of unclear project scope, size, etc.
- Nevertheless, it is one of their critical tasks that they have to finish and provide the requested information.
- The best solution is to find a relatively objective method of effort estimation and provide the requested information.

Software Testing

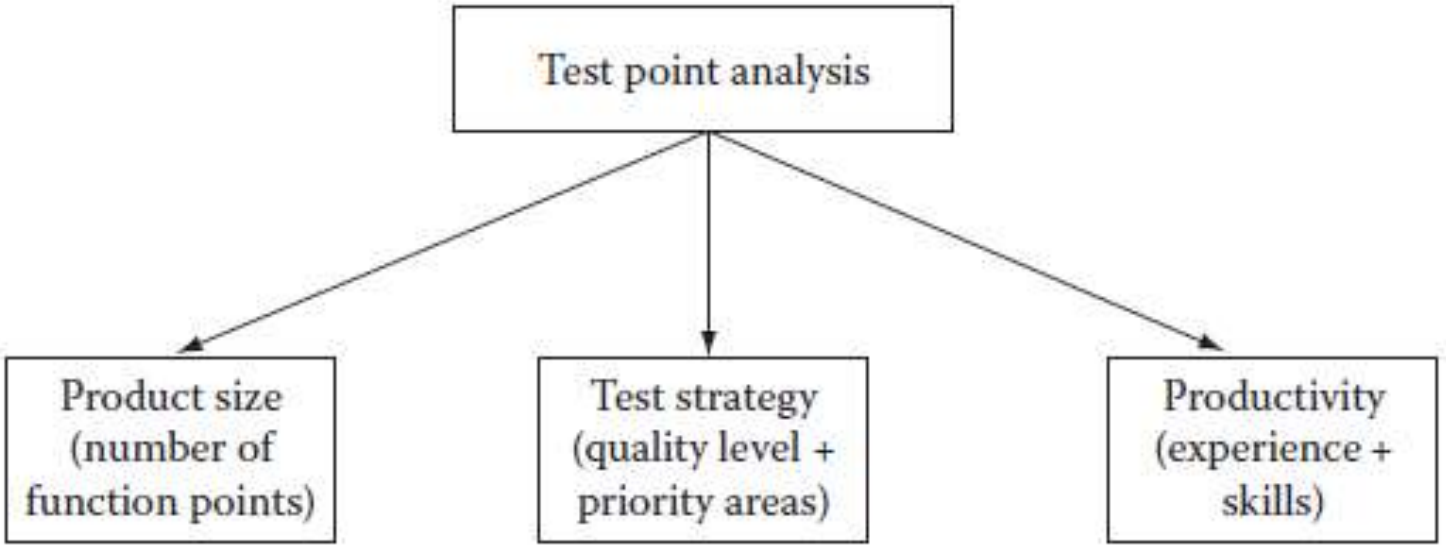
Test Strategy and Planning – Effort Estimation: Test Point Analysis

- There are many methods available for effort estimation for test projects.
- Some of them include test point analysis, the wideband Delphi technique, experience-based estimation, etc.
- In the test point analysis technique, three inputs required are project size, test strategy, and productivity.
- Project size is determined by calculating the number of test points in the software application which is being developed.
- Test points, in turn, are calculated from function points.
- The number of function points is calculated from the number of functions and function complexity.
- If the number of function points in the application has been calculated by the development team, then test points are calculated from the available function point information.
- Otherwise rough function point data can be used (Figure below - Test point analysis components).
- A test strategy is derived from two pieces of information from the customer, what will be the quality level for the application and which features of the application will be used most frequently.

Software Testing

Test Strategy and Planning – Effort Estimation: Test Point Analysis

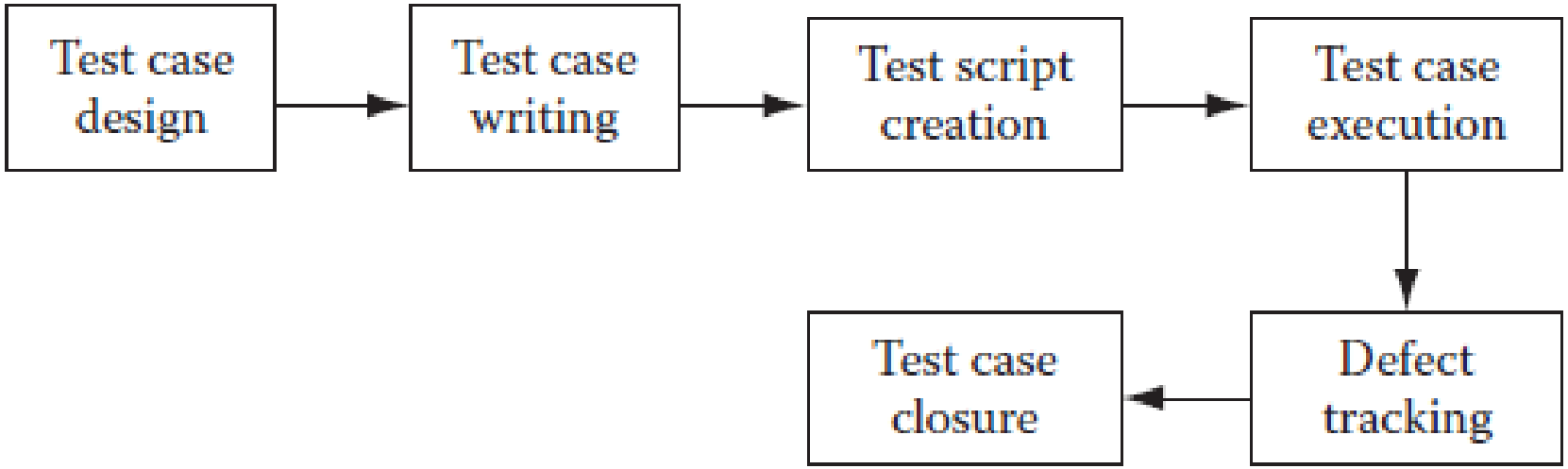
- Productivity is derived from knowledge and experience of the test team members.
- While productivity can be calculated objectively without taking reference from any statistical data, it makes sense to use past productivity data from previously executed projects to make productivity figures more realistic.
- In case of iterative development, testing cycles will be short and iterative in nature.
- The test manager should make the test effort calculations accordingly.



Software Testing

Test Project Monitoring and Control

- Test projects involve a large variety of activities including test case design, test case management, test case automation, test execution, defect tracking, verifying and validating the application under test, etc (Figure below – Test life cycle).



Test Case Design

- A proper test case design plan goes a long way in ensuring that test cases are designed properly.
- The test manager has to ensure which kind of tests are to be designed, how many test cases have to be written for particular modules and which test areas are priority areas.

Software Testing

Test Project Monitoring and Control – Test Case Design: Test Types

- An application may have to be tested for functionality, performance, usability, compatibility and many other kinds of things to make sure it is really useful for end users.
- For each kind of testing, a set of test cases has to be written and executed then finally, the system should be verified and validated.
- For applications that have many versions, regression tests also have to be performed.
- Managing all these kinds of testing is a big task for the test manager.
- A good test manager will first divide the testing tasks on the basis of test types.
- Then tasks can be further divided by modules.
- After that, he can allocate testing tasks to test engineers appropriately.
- There is one more way of segregating tests.
- Depending on the project phase, we need to perform system testing, integration testing or user acceptance testing.
- Usually when the application is built after the construction phase, it has to be tested and verified whether it is functioning as per requirements.

Software Testing

Test Project Monitoring and Control – Test Case Design: Test Types

- Integration testing is performed when the application needs to be integrated with any other external application to ensure that integration is proper.
- User acceptance testing is done by end users.
- If any defect is found during these tests, they are fixed so that the application goes into production with as few defects as possible.

Test Project Monitoring and Control – Test Case Management

- There could be existing test cases as well as new test cases that also need to be created.
- Test case management involves managing different versions of test cases, keeping track of changes in them, keeping a separate repository of test cases based on type of tests, as well as creating and managing automation scripts.

Software Testing

Test Project Monitoring and Control – Test Bed Preparation

- Test bed preparation involves installing the application on a machine that is accessible to all test teams.
- Care is taken to ensure that this machine is free of any interference from unauthorized access.
- Test data is populated in the application.
- Care should also be taken to ensure that the test bed resembles the production environment as closely as possible, including all software and hardware configurations.
- For all types of testing, it is very important that the “application under test” (UAT) should be tested under an environment that is as close to the environment under which the proposed application will be deployed for production.
- That is why test bed preparation is very important.
- The application should be installed on a dedicated server that has the same configuration as the proposed production environment.
- This server should not be used for any other purpose except for testing.

Software Testing

Test Project Monitoring and Control – Test Bed Preparation

- It should be installed centrally, so that even distributed teams, contractors or service providers can easily access it using remote desktop sharing or any peer to peer networking protocol over the Internet.
- If the application can be directly accessed over the Internet then it is even better.
- There should not be any testing done on applications that are deployed on the local test engineer's machine.
- To gain familiarity with the application and preliminary testing, it is acceptable to have a local copy of the application, but never for testing when defects are to be logged and verified by many people.
- It is because it is very important to reproduce the defect when the developer or any concerned person asks for it.
- In case of disputes, if a defect cannot be reproduced, then it becomes difficult for the test team to justify why a defect has been logged when others cannot reproduce it.
- That is the reason for which the test bed should be prepared very carefully and kept as isolated from any other environment as much as possible to preserve its integrity.

Software Testing

Test Project Monitoring and Control – Test Bed Preparation

- The test data preparation is also a very tricky affair.
- The test data should closely resemble what the end users use in their daily transactions.
- For this, the test team can get some business data already used by the end users.
- The test bed should be populated with a similar kind of data.

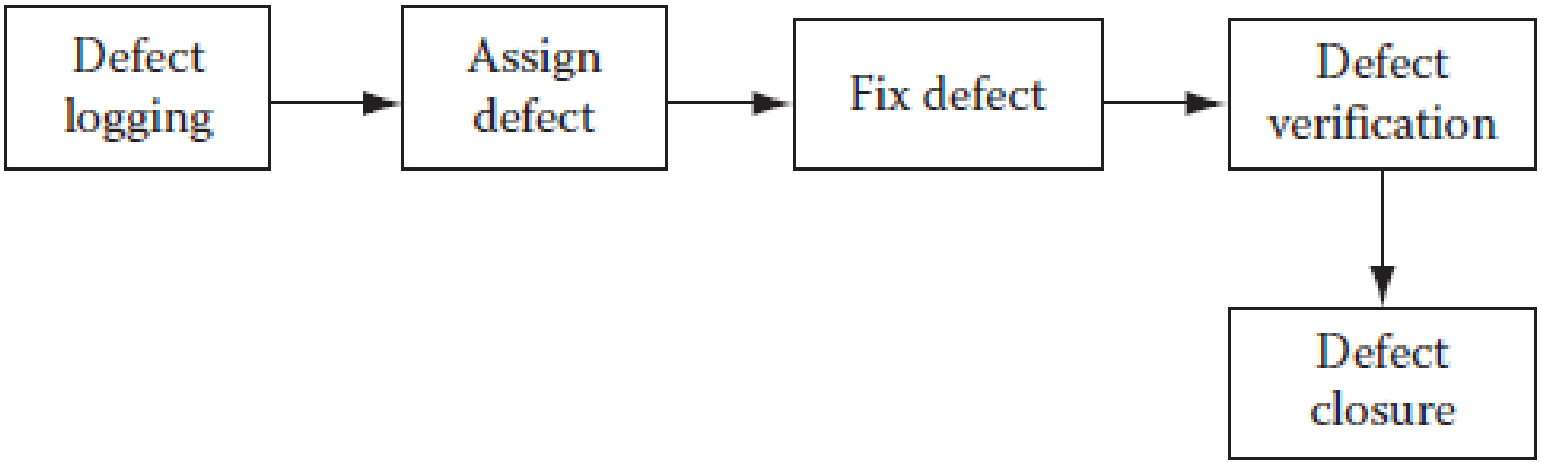
Test Project Monitoring and Control – Test Case Execution

- Test case execution involves executing prepared test cases manually or using automation tools to execute them.
- For regression tests, automated test execution is a preferred method.
- After each test case is executed, it may pass or fail.
- If it fails then defects have to be logged.
- Exit criteria for test case execution cycle are generally defined in advance.
- Generally, when a certain level of quality of the application is reached then test execution stops.

Software Testing

Test Project Monitoring and Control – Defect Tracking

- Defect tracking is one of the most important activities in a test project.
- During defect tracking it is ensured that defects are logged and get fixed.
- All defects and their fixing are tracked carefully (Figure below – Defect life cycle).
- Defect count per hour per day is a common way of measuring performance of a test team.



- If the testing is done for an in-house software product, traditionally, it is used to, not be a performance evaluation measurement.
- What really counted was the number of defects found in production when the software product was deployed and used by end users.
- But it is too late for a performance measurement.

Software Testing

Test Project Monitoring and Control – Defect Tracking

- What if many of the test team members left before the product was deployed?
- In fact this is a reality, given the high attrition rate (as much as 20% at many corporations) of software professionals.
- Once they are gone, there is no point in measuring the performance.
- Thus, a better measurement would allow for more immediate results.
- This is achieved by measuring the defect count per hour per day.
- Then there is the case of outsourced test projects.
- If the contract is only for testing up to deployment and not afterward, then measurement does not make sense after the contract has ended.
- A good defect tracking application should be deployed on a central server that is accessible to all test and development teams.
- Each defect should be logged in such a way that it could be understood by both development and testing teams.
- Generally, the defects should be reproducible, but in many instances, this is difficult.
- In such instances, a good resolution should be made by the test and development managers.

Software Testing

Test Case Reporting

- During the execution of a test project, many initial and final reports are made.
- But status reports also need to be made.
- Test reports include test planning reports, test strategy reports, requirement document review comments, number of test cases created, automation scripts created, test execution cycle reports, defect tracking reports, etc.
- Some other reports include trace-ability matrix reports, defect density, test execution rate, test creation rate, test automation script writing rate, etc.

REFERENCES

- Ashfaque Ahmed, Software Project Management: A Process-driven approach, Boca Raton, Fla: CRC Press, 2012.

THANK YOU