COURSE OUTCOMES

Apply Black Box, White Box, and Selenium testing techniques in commercial environment for improving the quality of software product.
Choose the techniques and skills for testing software projects using modern software testing tools
Analyse V&V activities, software testing life cycle and methodologies, test automation life cycle and its frame work.
Create test cases for manual and automation testing.

INTRODUCTION

- Software testing is an established process
- Effective testing , not exhaustive testing
- Testing should be done with the intention of finding more and more bugs
- It is not a single phase in SDLC, but it starts as soon as the requirements in a project have been gathered.
- STLC is established in parallel to SDLC.

Evolution of software testing

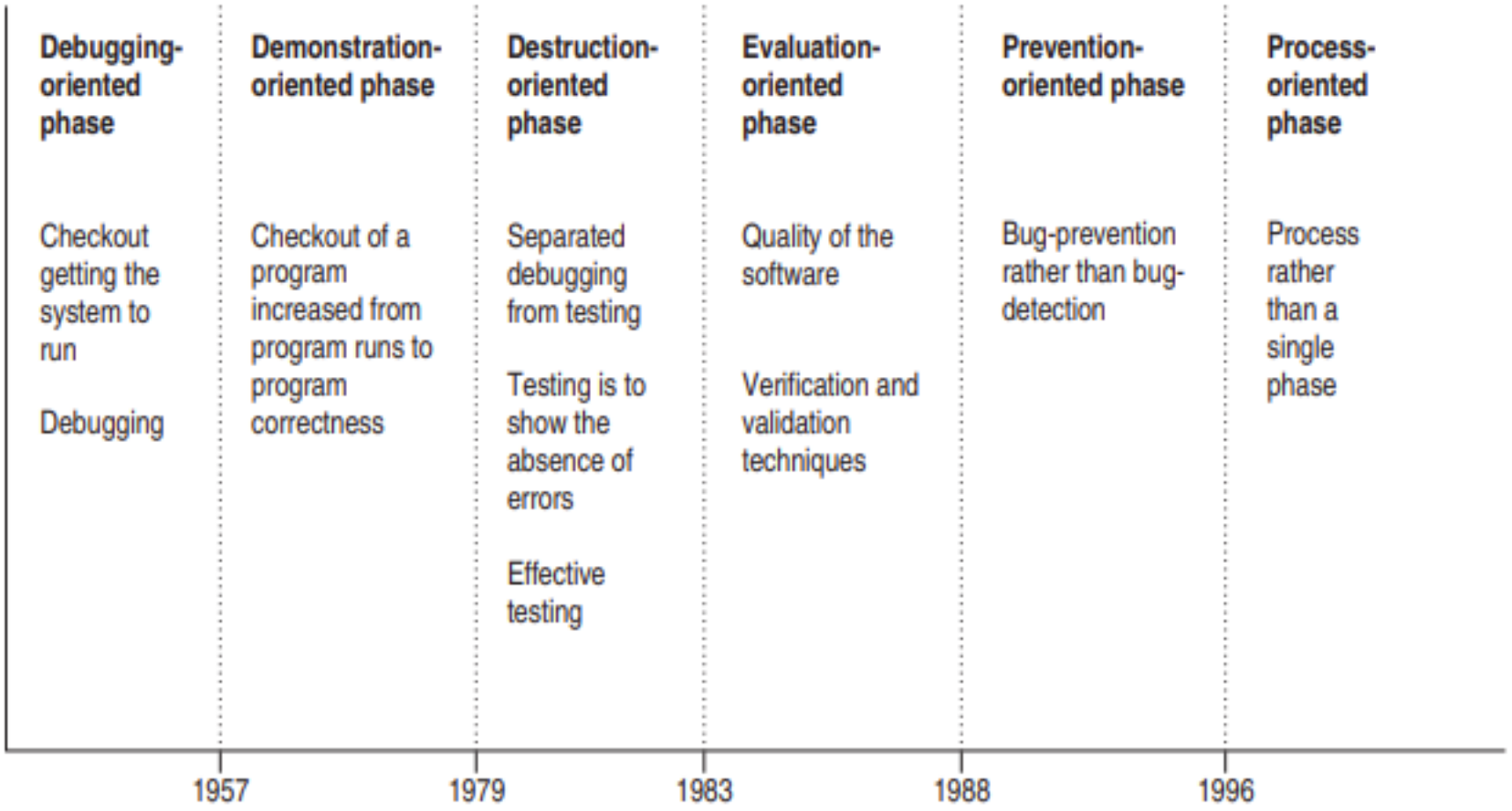


Figure 1.1 Evolution phases of software testing

Debugging oriented Phase (before 1957)

- This phase is early period of testing
- Testing basics were unknown
- Programs were written and then tested by the programmers until the bugs are removed.
- Checkout is the term used for testing
- No clear distinction between software development, testing and debugging.

Demonstration oriented phase (1957-78)

- The term debugging is continued
- The purpose of checkout is not only to run the software but also to demonstrate the correctness as per the requirements.
- Scope of checkout is increased from program runs to program correctness.
- Purpose is to show the absence of errors.
- No stress on the test case design
- Misconception – software could be tested exhaustively.

Destruction-oriented Phase (1979–82)

- The view of testing has been changed from testing is to show the absence of errors' to 'testing is to find more and more errors.
- Debugging is separated from testing and focused on valuable test cases
- Importance is given to effective testing in comparison to exhaustive testing
- Realized that testing is to be started from the beginning.

Evaluation-oriented Phase (1983–87)

- Stresses on the quality of software products
- Early testing concept was established in the form of verification and validation activities which help in producing better quality software.

Prevention-oriented Phase (1988–95)

- Evaluation model Stressed on the concept of bug-prevention as compared to the earlier concept of bug-detection
- With the idea of early detection of bugs in earlier phases, we can prevent the bugs in implementation or further phases.
- bugs can also be prevented in other projects with the experience gained in similar software projects.
- The prevention model includes test planning, test analysis, and test design activities playing a major role
- Evaluation model mainly relies on analysis and reviewing techniques other than testing

Process-oriented Phase (1996 onwards)

- Testing was established as a complete process rather than a single phase in SDLC
- Testing process starts as soon as the requirements for a project are specified and it runs parallel to SDLC
- The performance of a testing process is done through TMM(Testing Maturity Model)
- Focus on quantification of various parameters which decide the performance of a testing process.

- **Software testing in three phases.**

Software Testing 1.0

- Software testing is considered as a single phase to be performed after coding of the software in SDLC.
- No test organization
- Very few testing tools were present and usage is very much limited due to high cost
- Management was not concerned with testing, as there was no quality goal.

Software Testing 2.0

- software testing gained importance in SDLC
- concept of early testing started
- Testing was evolving in the direction of planning the test resources.
- Many testing tools were also available in this phase

Software Testing 3.0

- Software testing is being evolved in the form of a process which is based on strategic effort.
- It gives us a roadmap of the overall testing process
- It should be driven by quality goals so that all controlling and monitoring activities can be performed by the managers
- Management is actively involved in this phase

SOFTWARE TESTING—MYTHS AND FACTS

- Myth : Testing is a single phase in SDLC
- Truth : It is a myth, that software testing is just a phase in SDLC and we perform testing only when the running code of the module is ready.
- But in reality, testing starts as soon as we get the requirement specifications for the software.
- Testing work continues throughout the SDLC, even post-implementation of the software

Myth : Testing is easy.

Truth : This myth is more in the minds of students who have just passed out or are going to pass out of college and want to start a career in testing.

- So the general perception is that, software testing is an easy job, wherein test cases are executed with testing tools only.
- But in reality, tools are there to automate the tasks and not to carry out all testing activities.
- Testers' job is not easy, as they have to plan and develop the test cases manually and it requires a thorough understanding of the project being developed with its overall design.
- Overall, testers have to shoulder a lot of responsibility which sometimes make their job even harder than that of a developer

Myth : Software development is worth more than testing.

Truth: This myth prevails in the minds of every team member and even in fresher's who are seeking jobs.

- we have this myth right from the beginning of our career, and testing is considered a secondary job.
- But testing has now become an established path for job-seekers
- Testing is a complete process like development, so the testing team enjoys equal status and importance as the development team

Myth : Complete testing is possible

Truth : This myth also exists at various levels of the development team.

- Complete testing at the surface level assumes that if we are giving all the inputs to the software, then it must be tested for all of them
- But in reality, it is not possible to provide all the possible inputs to test the software, as the input domain of even a small program is too large to test.
- there are many things which cannot be tested completely, as it may take years to do so
- This is the reason why the term ‘complete testing’ has been replaced with ‘effective testing
- Effective testing is to select and run some select test cases such that severe bugs are uncovered first

- **Myth** : Testing starts after program development.
  Truth : Most of the team members, who are not aware of testing as a process, still feel that testing cannot commence before coding.
- But this is not true, the work of a tester begins as soon as we get the specifications.
- The tester performs testing at the end of every phase of SDLC in the form of verification and plans for the validation testing .
- He writes detailed test cases, executes the test cases, reports the test results, etc.
- Testing after coding is just a part of all the testing activities.

- **Myth** : The purpose of testing is to check the functionality of the software.
- **Truth :** Today, all the testing activities are driven by quality goals.
- Ultimately, the goal of testing is also to ensure quality of the software.
- But quality does not imply checking only the functionalities of all the modules.
- There are various things related to quality of the software, for which test cases must be executed

- **Myth :** Anyone can be a tester.
- **Truth :** This is the extension of the myth that 'testing is easy.'
- Most of us think that testing is an intuitive process and it can be performed easily without any training. And therefore, anyone can be a tester.
- As an established process, software testing as a career also needs training for various purposes, such as to understand (i) various phases of software testing life cycle, (ii) recent techniques to design test cases, (iii) various tools and how to work on them, etc.
- This is the reason that various testing courses for certified testers are being run.

Software Testing – Goals

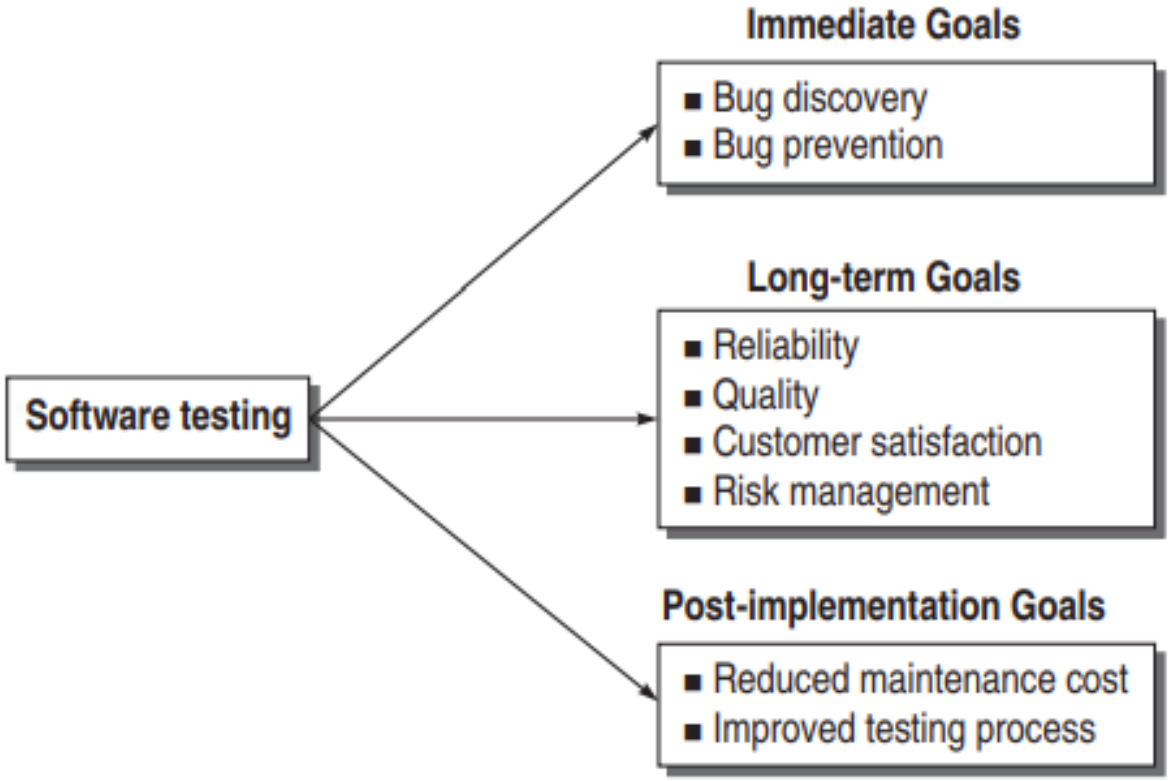


Figure 1.2 Software testing goals

Short-term or immediate goals

These goals are the immediate results after performing testing. These goals may be set in the individual phases of SDLC

Bug discovery : The immediate goal of testing is to find errors at any stage of software development.

More the bugs discovered at an early stage, better will be the success rate of software testing.

- **Bug prevention** : It is the consequent action of bug discovery.
- Software development team members gets to learn how to code safely such that the bugs discovered should not be repeated in later stages or future projects.
- Though errors cannot be prevented to zero, they can be minimized.
- Bug prevention is a superior goal of testing

Long-term goals

These goals affect the product quality in the long run, when one cycle of the SDLC is over.

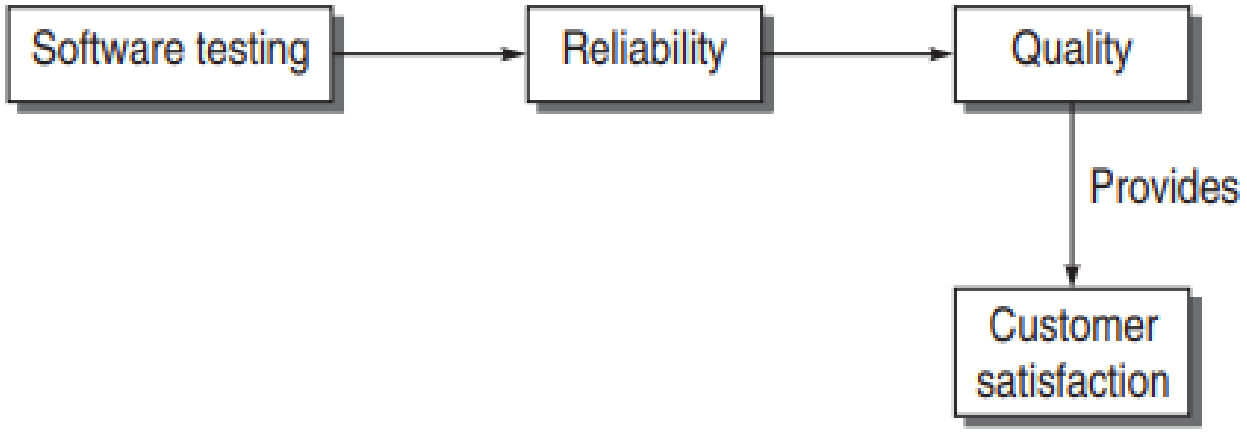
- **Quality** : Since software is also a product, its quality is primary from the users' point of view.
- Thorough testing ensures superior quality.
- Therefore, the first goal of understanding and performing the testing process is to enhance the quality of the software product.
- Though quality depends on various factors, such as correctness, integrity, efficiency, etc., reliability is the major factor to achieve quality.

- The software should be passed through a rigorous reliability analysis to attain high quality standards.
- Reliability is a matter of confidence that the software will not fail, and this level of confidence increases with rigorous testing.
- The confidence in reliability, in turn, increases the quality, as shown in Fig. below

Customer satisfaction

- Testing should be complete in the sense that it must satisfy the user for all the specified requirements mentioned in the user manual, as well as for the unspecified requirements which are otherwise understood

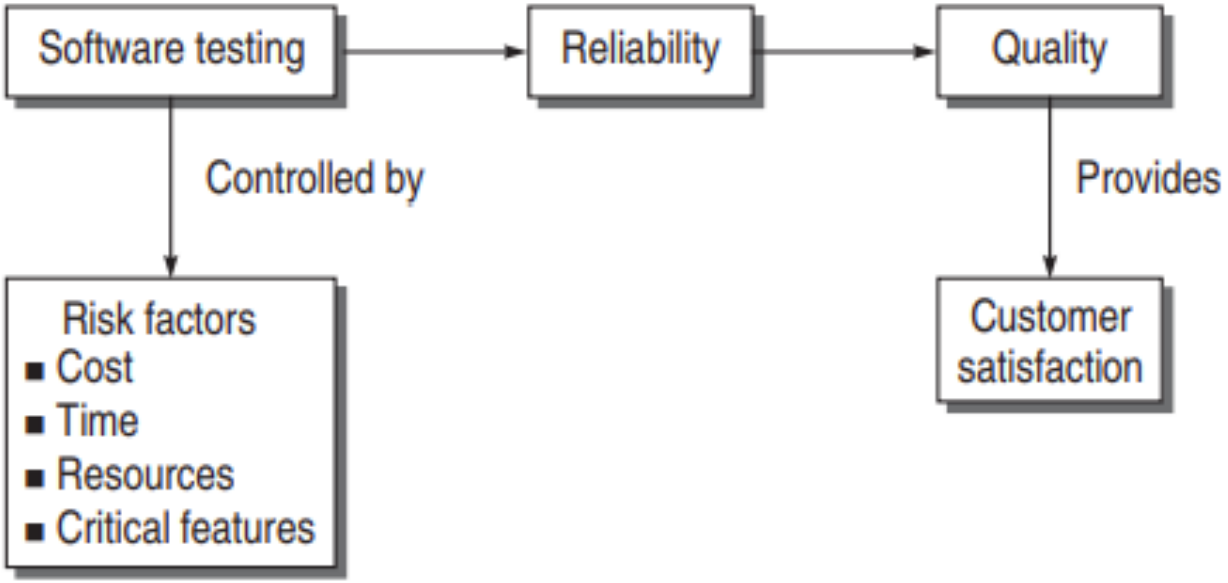
- A complete testing process achieves reliability, reliability enhances the quality, and quality in turn, increases the customer satisfaction, as shown in Fig below



Risk Management

- Risk is the probability that undesirable events will occur in a system.
- risk is basically concerned with the business perspective of an organization.
- Risks must be controlled to manage them with ease

- It is the testers responsibility to evaluate business risks
- Testers should also categorize the levels of risks after their assessment.
- Risk management becomes the long term goal



Post-implementation goals

- These goals are important after the product is released
- Reduced maintenance cost : The only maintenance cost in a software product is its failure due to errors
- Post-release errors are costlier to fix, as they are difficult to detect
- The maintenance cost is reduced if testing has been done rigorously and effectively,

Improved software testing process

- A testing process for one project may not be successful and there may be scope for improvement.
- The bug history and post-implementation results can be analyzed to find out snags in the present testing process, which can be rectified in future projects
- The long-term post-implementation goal is to improve the testing process for future projects.

Software Testing Definitions

- Testing is the process of executing a program with the intent of finding errors.
- A successful test is one that uncovers an as-yet-undiscovered error.
- Testing can show the presence of bugs but never their absence.
- Testing is a support function that helps developers look good by finding their mistake before anyone else does.

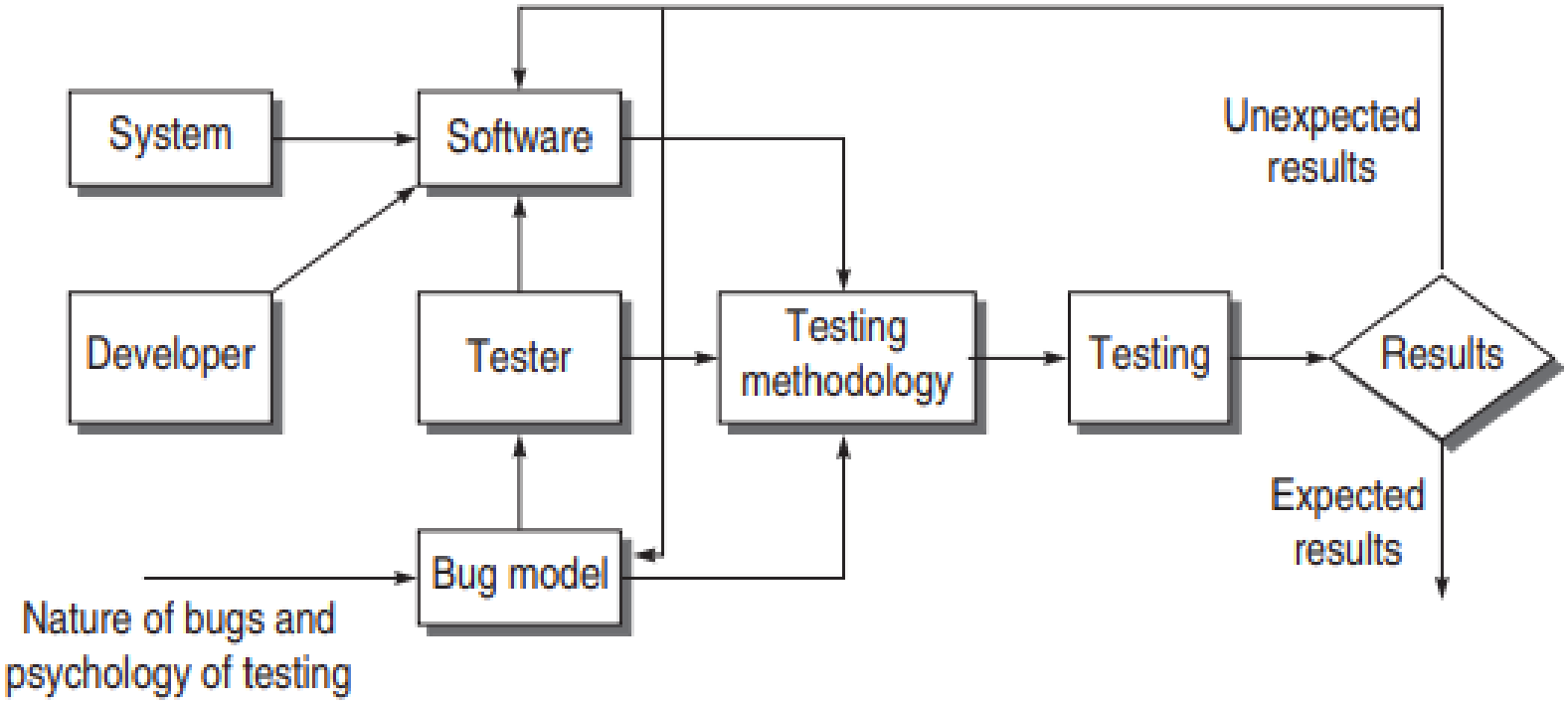
- Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate.
- Program testing is to affirm software quality with methods that can be economically and effectively applied to both large-scale and small-scale systems.
- Testing is a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of the software being tested.
- Software testing is a process that detects important bugs with the objective of having better quality software.

Model for Software Testing

- Software is basically a part of a system for which it is being developed.
- Systems consist of hardware and software to make the product run
- Developer develops the software in the prescribed system environment considering the testability of the software
- Testers are supposed to get on with their tasks as soon as the requirements are specified

- Testers work on the basis of a bug model which classifies the bugs based on the criticality or the SDLC phase in which the testing is to be performed.
- Based on the software type and the bug model, testers decide a testing methodology which guides how the testing will be performed
- If the testing results are in line with the desired goals, then the testing is successful; otherwise, the software or the bug model or the testing methodology has to be modified, so that the desired results are achieved.

SOFTWARE TESTING MODEL



Software and Software Model

- Software is built after analysing the system in the environment
- It is a complex entity which deals with environment, logic, programmer psychology, etc.
- Complex software makes it very difficult to test
- Developers should design and code the software such that it is testable at every point.
- The software to be tested may be modeled such that it is testable, avoiding unnecessary complexities

Bug Model

- Bug model provides a perception of the kind of bugs expected
- Considering the nature of all types of bugs, a bug model can be prepared that may help in deciding a testing strategy.
- However, every type of bug cannot be predicted
- If we get incorrect results, the bug model needs to be modified

Testing methodology and Testing

- Based on the inputs from the software model and the bug model, testers can develop a testing methodology that incorporates both testing strategy and testing tactics.
- Testing strategy is the roadmap that gives us well-defined steps for the overall testing process.
- It prepares the planned steps based on the risk factors and the testing phase

- software testing techniques and testing tools can be applied within these steps
- However, if we don’t get the required results, the testing plans must be checked and modified accordingly

EFFECTIVE SOFTWARE TESTING VS. EXHAUSTIVE SOFTWARE TESTING

- Exhaustive or complete software testing means that every statement in the program and every possible path combination with every possible combination of data must be executed.
- Exhaustive or complete testing is not possible.
- Effective testing which emphasizes efficient techniques to test the software so that important features will be tested within the constrained resources

- The testing process should be understood as a domain of possible tests There are subsets of these possible tests. But the domain of possible tests becomes infinite, as we cannot test every possible combination.
- Computer speed and time constraints limit the possibility of performing all the tests
- Complete testing requires the organization to invest a long time which is not cost-effective.
- Testing must be performed on selected subsets that can be performed within the constrained resources

- selected group of subsets, but not the whole domain of testing, makes effective software testing
- Effective testing can be enhanced if subsets are selected based on the factors which are required in a particular environment

Domain of Possible Inputs to the Software

- The domain of input data has four sub-parts:

(a) valid inputs,

(b) invalid inputs,

(c) edited inputs, and

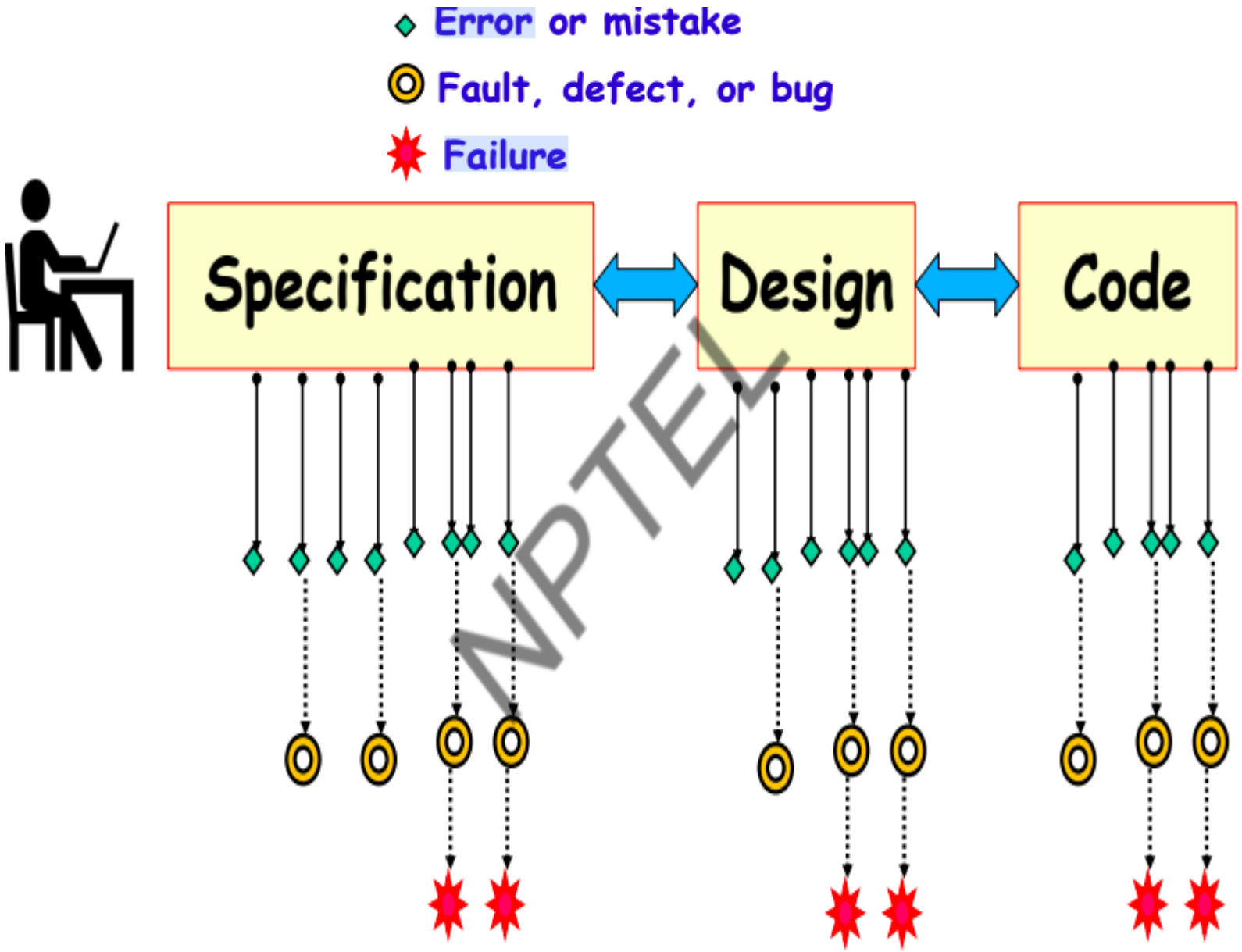
(d) race condition inputs

- Valid Inputs:
- test every valid input on the software
- How can we test all possible combinations?
- Ex: adding two-digit two numbers. Their range is from –99 to 99 (total 199). So the total number of test case combinations will be 199 × 199 = 39601.

INTRODUCTION

A program may fail during testing.

- A manifestation of a fault also called defect or bug
- Presence of a fault may not lead to a failure.


Error or mistake
Fault, defect, or bug
Failure
Specification
Design
Code