SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

Introduction

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Context : Some Quotes on Testing

"Testing is the process of executing a program with the intention of finding errors."

– Myers

"Testing can show the presence of bugs but never their absence."

– Dijkstra

"Good testing at the minimum is as difficult as good design"

Testing is executing software in a simulated or real environment, using inputs selected *somehow*.

Testing does NOT guarantee a completely defect free product

SOFTWARE TESTING

Why test

Inadequate or Incomplete testing can lead to

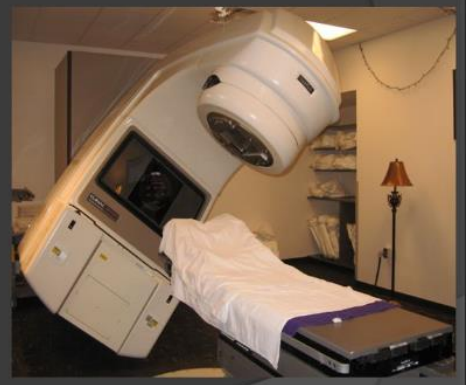
Therac-25 Medical Accelerator

1985-1987

Radiation therapy device malfunctions, delivers lethal doses at several facilities

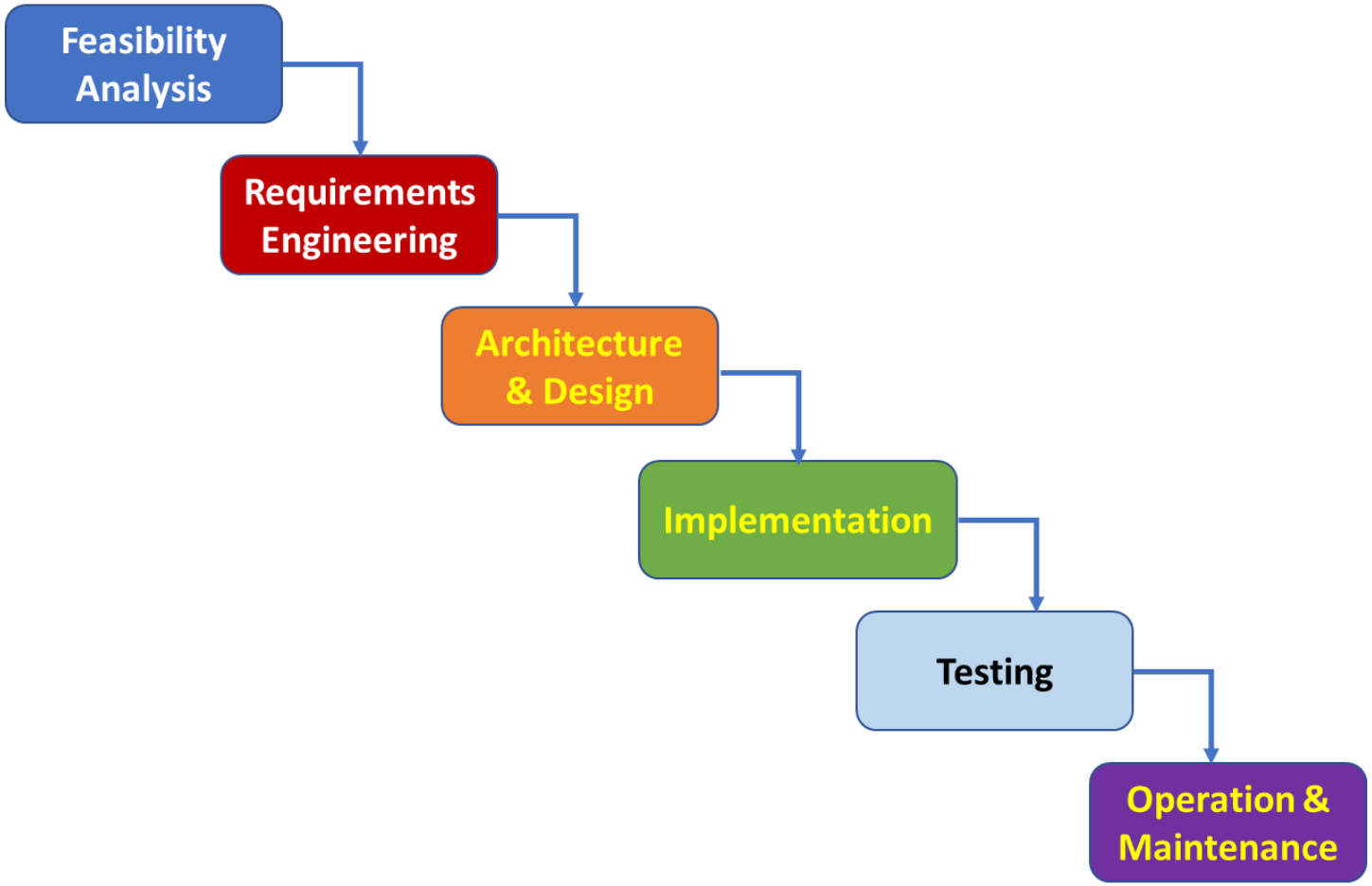
The 25 was an improved version of an older model

It could deliver beta-particles (electron beam) or x-rays



Generics

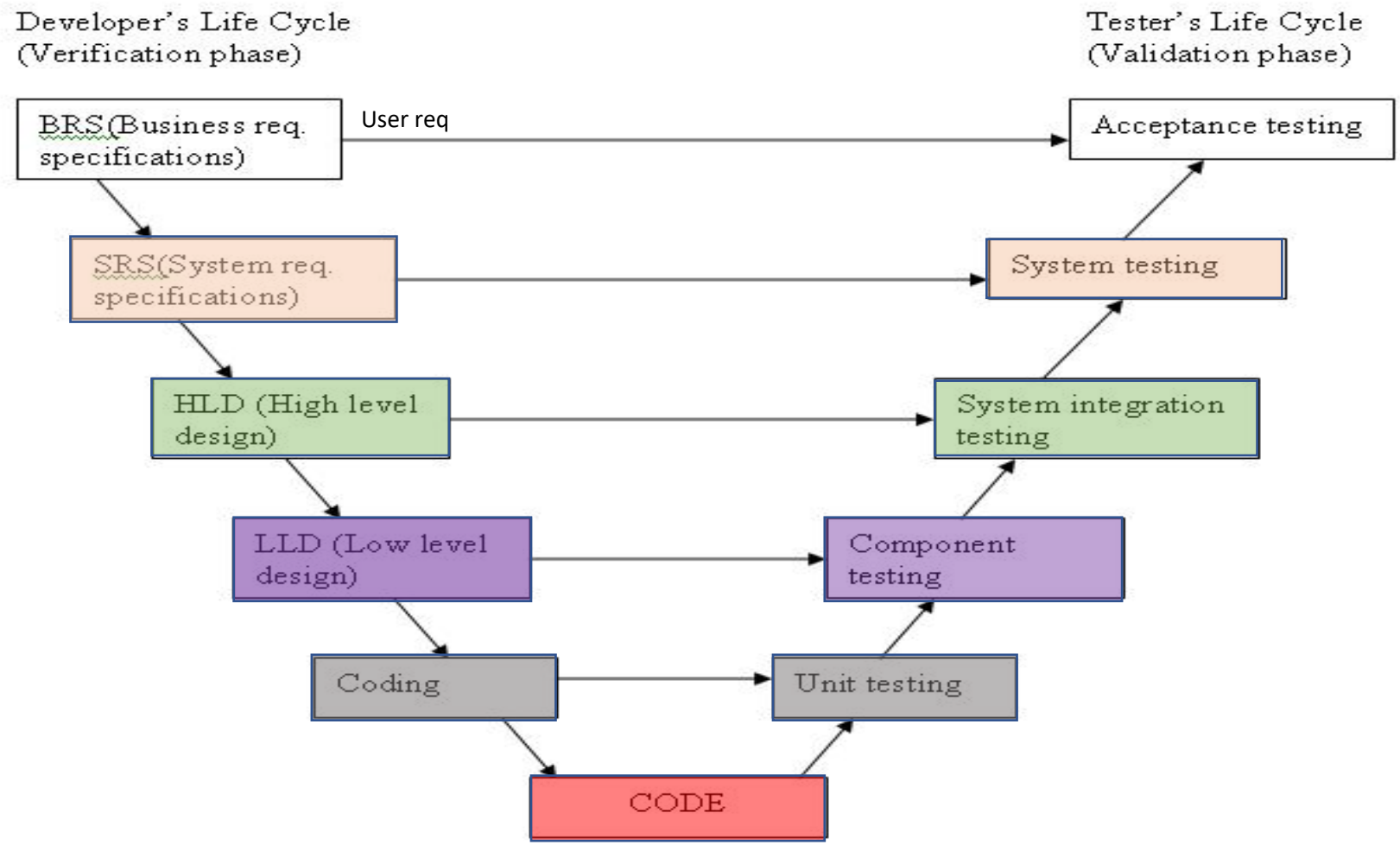
- In an typical plan driven SDLC, Testing is a phase which follows Requirements, Architecture, Design and Implementation.



SOFTWARE TESTING

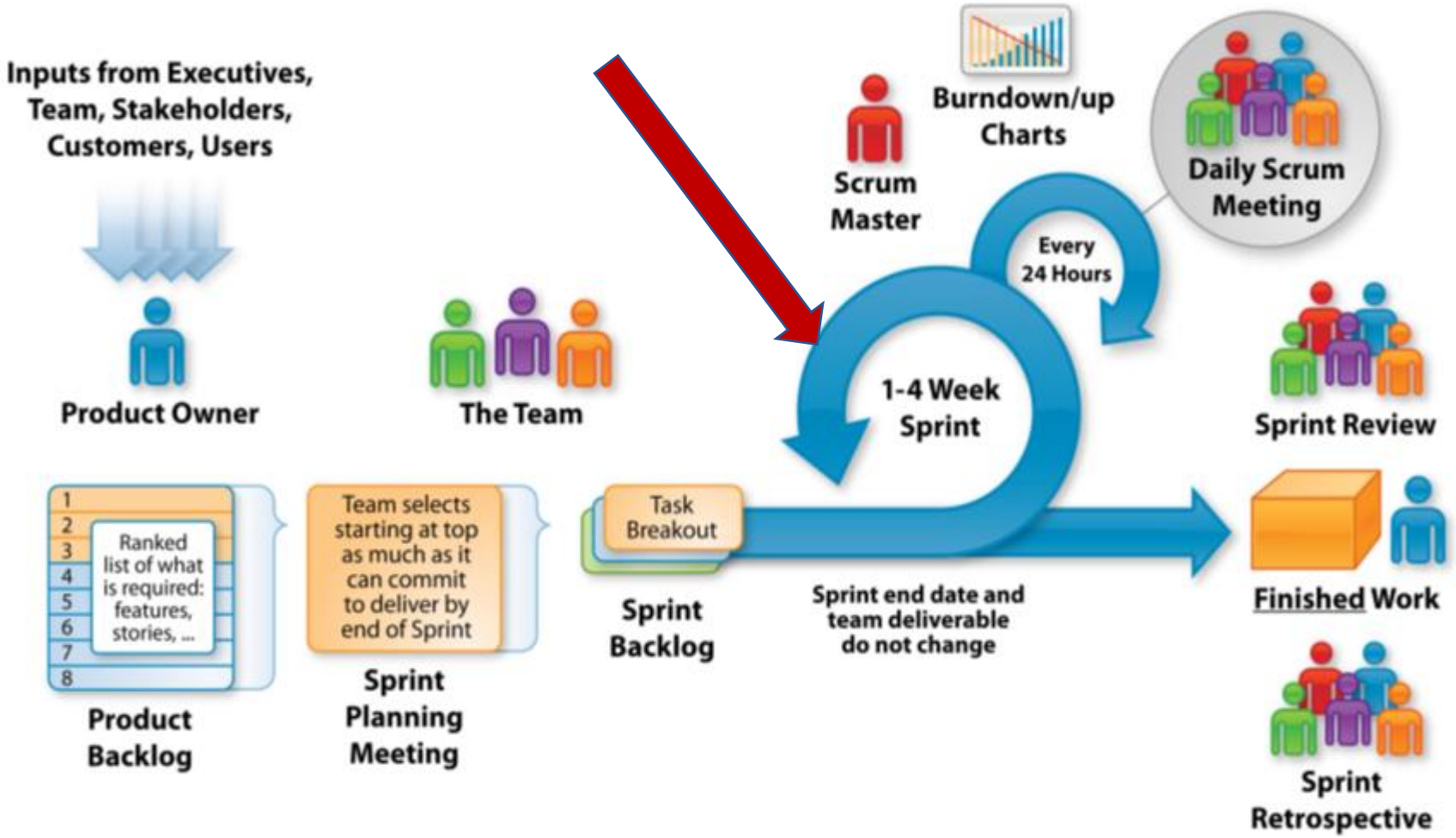
Generics

- If we consider this from a V Model perspective



Generics

- If we consider this from an Agile approach like SCRUM



SOFTWARE TESTING

Generics

- Testing as a phase is when the Verification and Validation of the Software is done. **Verification** is the process of evaluating a product to determine if ***it is built according to the*** specified requirements. **Validation** is the process of evaluating a system or component to determine if ***it satisfies specified requirements***.
- Testing establishes that the software developed, will perform as per the expectations specified in the requirements & exposes deviations if any
- Involves measuring attributes, such as performance or usability, estimating the operational reliability, and so on
- Testing builds confidence in the software that, it is good enough for its intended use. The planned usage of the software will determine the degree of confidence that is needed and the degree and type of testing.

SOFTWARE TESTING

Objectives of testing

Demonstration

- Show that the system can be used with acceptable risk
- Demonstrate functions under special conditions
- Show the products are ready for integration or use

Detection

- Discover defects, errors and deficiencies
- Determine system capabilities and limitations
- Determine quality of components, work products, and the system

Prevention

- Provide information to prevent or reduce the number of errors
- Reduce the number of early errors propagated through to later phases
- Clarify system specifications and performance
- Identify ways to avoid risks and problems in the future

SOFTWARE TESTING

VERIFICATION

"Are we building the product right”

- It’s the process of checking that the software achieves its goals to ensure that the products and deliverables meet specified requirements before final testing
- Typically does not include the execution of the code
- Verification finds the bugs early in the development cycle
- It is Static testing – includes checking documents, design, code (reviews, walkthroughs, Inspections ..)
- Verification process targets on software architecture, design, database, etc.
- Mostly happens before validation

SOFTWARE TESTING

VALIDATION

"Are we building the right product"

- Focuses on product related activities that attempt to determine if the system or project deliverables meet the customer or clients expectation
- Validation is predominantly done through Dynamic testing
  - Execution of code (Exercising and observing the behaviour)
  - Testing validates if the system has all the capabilities & features defined in the projects scope and requirements definition
  - Typically done by the Testing team
  - Its done again mostly after Verification
- Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

SOFTWARE TESTING

Terminologies : Defect, Bug, Failure, Issue

1. Defect

- Is a deviation from the requirements.
- A shortcoming/fault or imperfection found and may be due to a design mistake or a code mistake.

2. Bug

- Is typically a coding error that that prevents it from working as intended, or produces an incorrect result
- Typically a programmers mistake
- Typically found during testing

Terminologies : Defect, Bug, Failure, Issue

3. Failure

- Is the resulting state of a system due to a defect, where its unable to perform its function as expected
- occurs during the development lifecycle or later as a result of a defect, or a bug

4. Issue

- It is typically raised by the end user/customer when the product does not meet with expectation on functionality/performance etc.
- This is usually tracked.

Characterizing Testing

Testing always consists of observing a sample of executions, and giving a verdict over them.

Characterizing these observations

Why are we making these observations will be based on test objective (are we looking for faults, is it for determining that the release is fit for use, or for looking at usability)

How did we arrive at these test cases for execution would be based on the test strategy or approach which will be followed

Which of the samples should we observe would be based on the test selection (Random, Ad-hoc, using some algorithm, using statistics ….)

Random testing may not work most times if you want to find significant errors

Characterizing Testing

Characterizing these observations (Cont.)

How Much – How big a sample to be used for testing (coverage analysis or reliability measures)

Exhaustive testing most often is not feasible

What is it we execute – whole product or part of the product (Unit/Component …)

Where do we test – Specifically in Embedded system

When do we test – in the lifecycle

Besides, the whole of testing activity needs to be carried on according to a controlled formal procedure, requiring rigorous planning and documentation. This would be managed based on a plan,

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

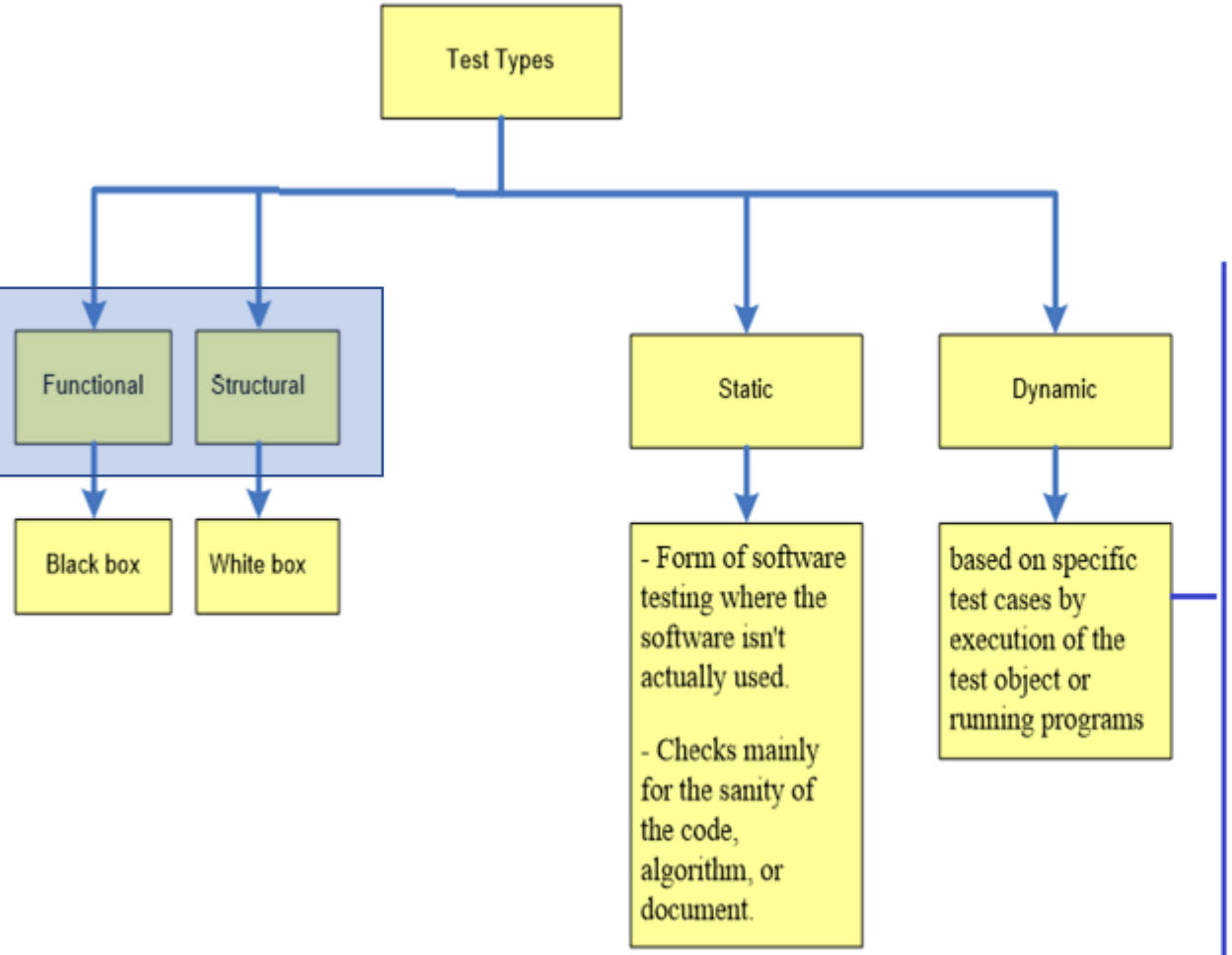
Testing Types

Dr. Phalachandra H. L
Department of Computer Science and Engineering

TESTING TYPES

BROAD CATEGORIZATION

Testing can be looked at from different perspectives



Levels (Lifecycle) of testing Types

- UNIT testing
- Integration teting
- System/Functional testing
- Acceptance Testing

Technique based testing type

- Coverage based testing
- Fault based testing

Manual

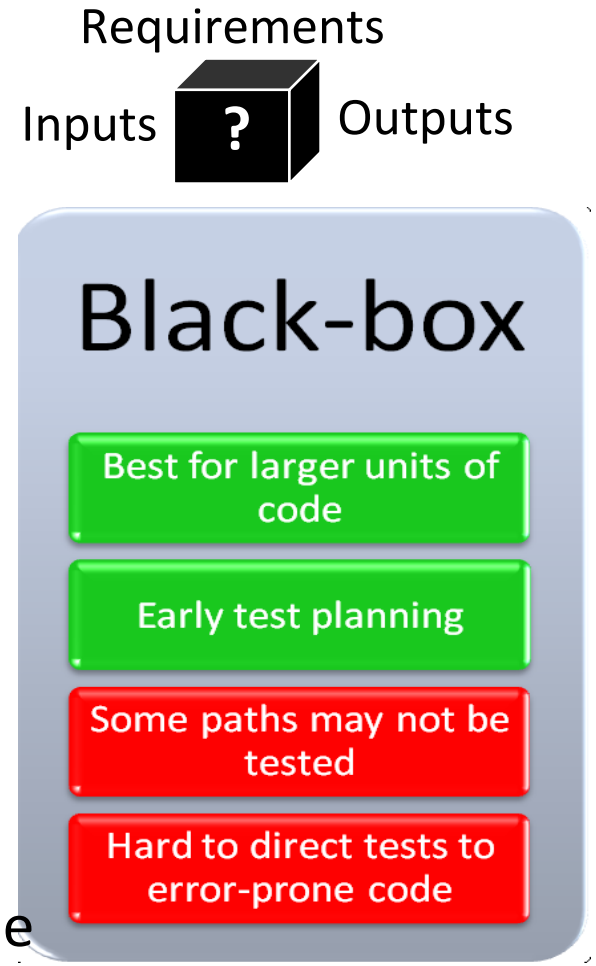
Automatic

TESTING TYPES

Block Box, White Box and Grey Box testing

Testing can be categorized based on the Functional or Structural approaches taken towards testing.

Black Box Testing (Functional)



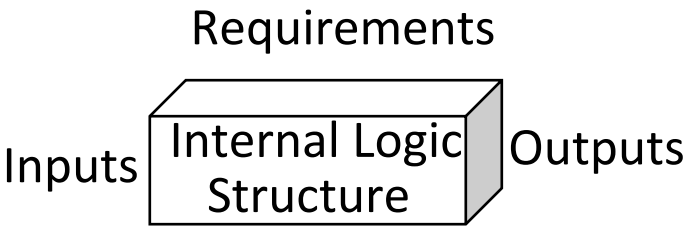
- Treats the software as a black box without regard to the internal structure or logic involved in the implementation.
- Concerned with the external behavior of the product.
- Objective is to identify defects in the output, which are generated as a result of valid and invalid input.
- It is testing from a user's point of view
- Typically test cases need to be provided to the tester, who then verifies that for a given input

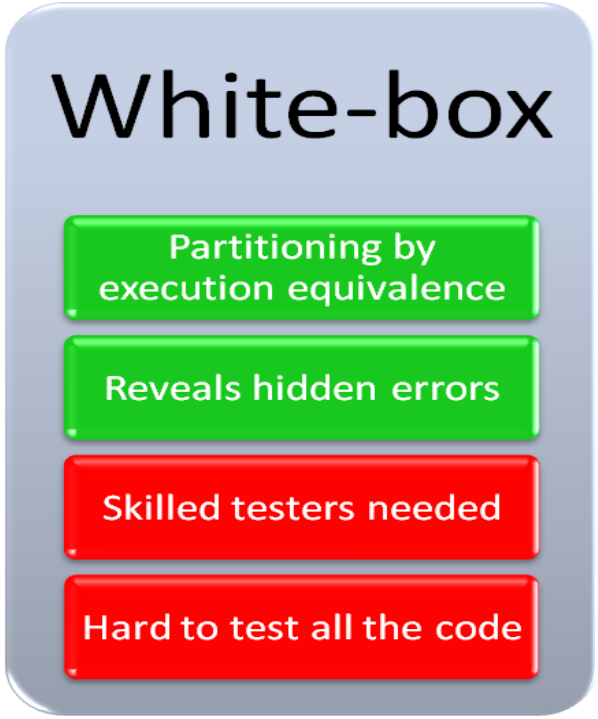
TESTING TYPES

Block Box, White Box and Grey Box testing

White Box Testing (Structural)

- Factors in the internal logic and structure of the code
- The test specifier uses knowledge of the internal structure of the software to derive test cases
- The test cases cannot be determined until the code has actually been written
- It is testing from a developer's point of view
- Typically needs testers to have programming skills



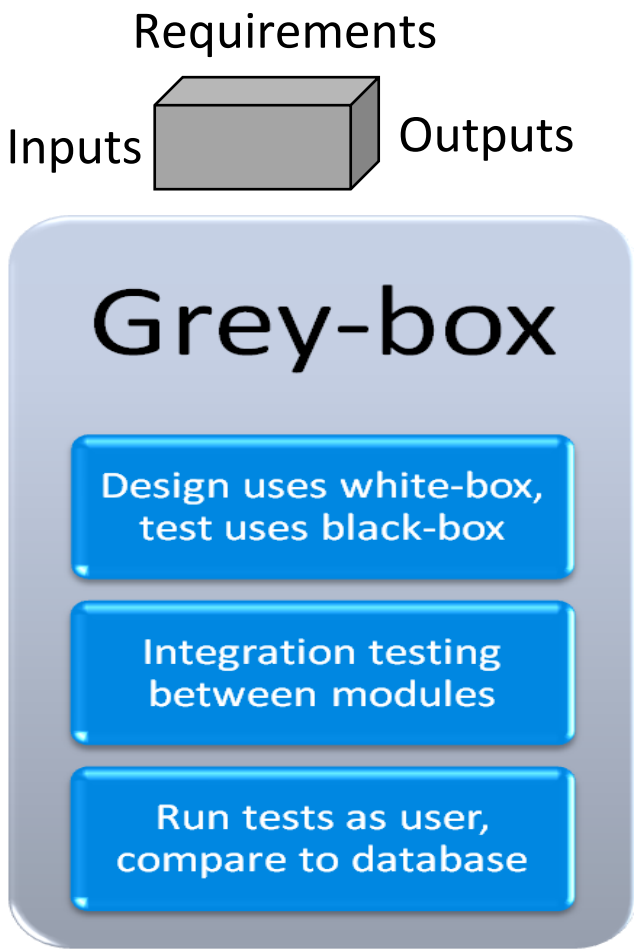


TESTING TYPES

Block Box, White Box and Grey Box testing (Cont.)

Grey Box (Hybrid) Testing (Mix of Functional & Structural)

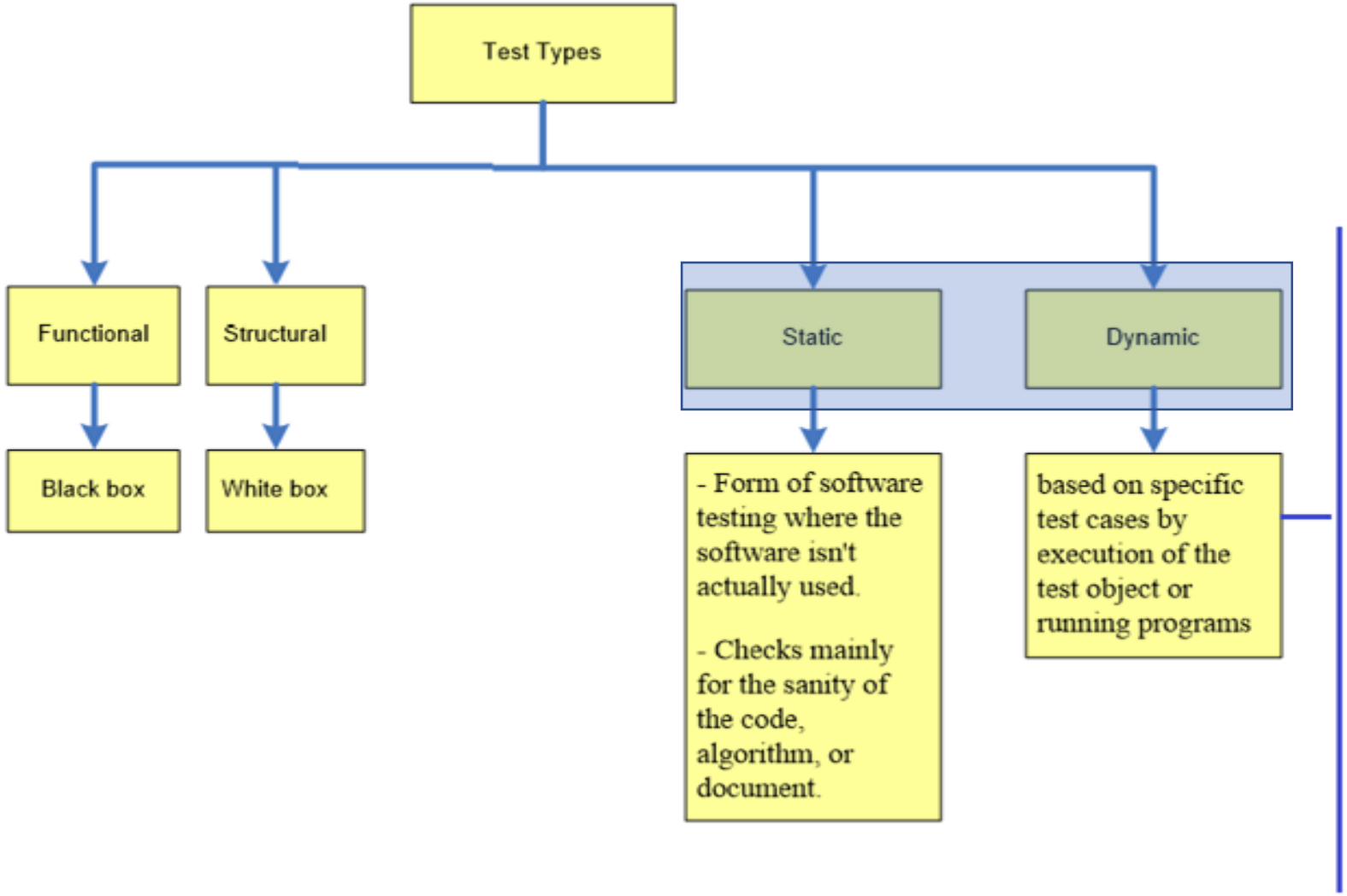
- Involves having access to internal data structures and algorithms for purposes of designing the test cases
- Testing is done at the user or black-box level



TESTING TYPES

Testing Types

Testing can also be looked at as Static and Dynamic based on the testing techniques used



Levels (Lifecycle) of testing Types

- UNIT testing
- Integration teting
- System/Functional testing
- Acceptance Testing

Technique based testing type

- Coverage based testing
- Fault based testing

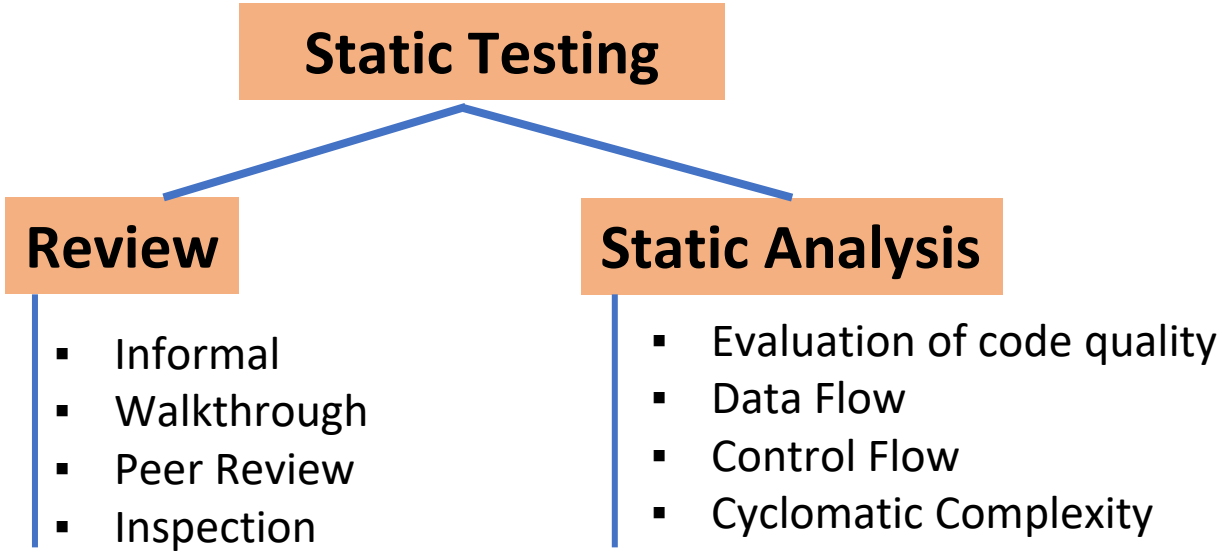
Manual

Automatic

Testing Types : Static and Dynamic Testing - 1

Static Testing

- In this approach you check for defects in the software without actually executing the code of the software/application.
- Static testing is performed in early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily



TESTING TYPES

Testing Types : Static and Dynamic Testing - 2

- Dynamic testing involves execution of the code for analyzing the dynamic behavior.
- This involves providing input values for the software under test, observation of the output values, which are then analyzed
- Has the advantage of being able to find difficult and complex defects which are not easily detectable by static testing
- Typically time and Budget consuming

There can be different kinds of dynamic testing like

1. Testing based on techniques like Code based or Fault based
2. Testing based on how the testing is done (Manual or Automatic)
3. Testing based on the levels of testing (will be discussed in the next session)

Testing Types : Testing Technique Based

Code Based

A control flow path is a graphical representation of all paths that might be traversed through a program during its execution

Control-flow-based criteria

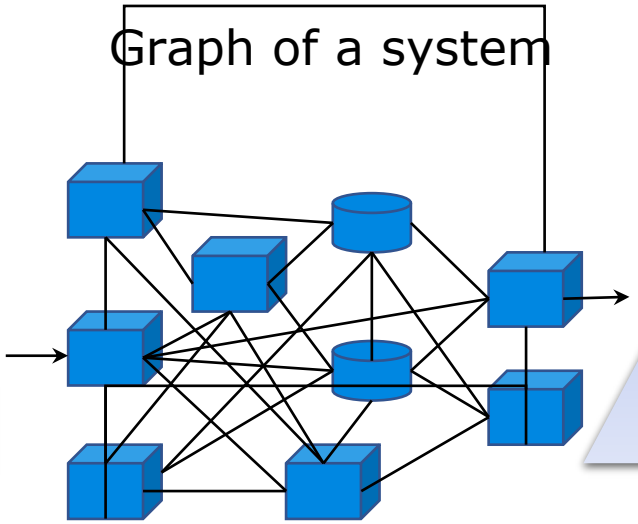
Path testing

Statement, branch, and condition testing

Branch coverage

Executes every path at least once

No branching leads to abnormal behavior?



Data-flow-based criteria

All definitions or all uses

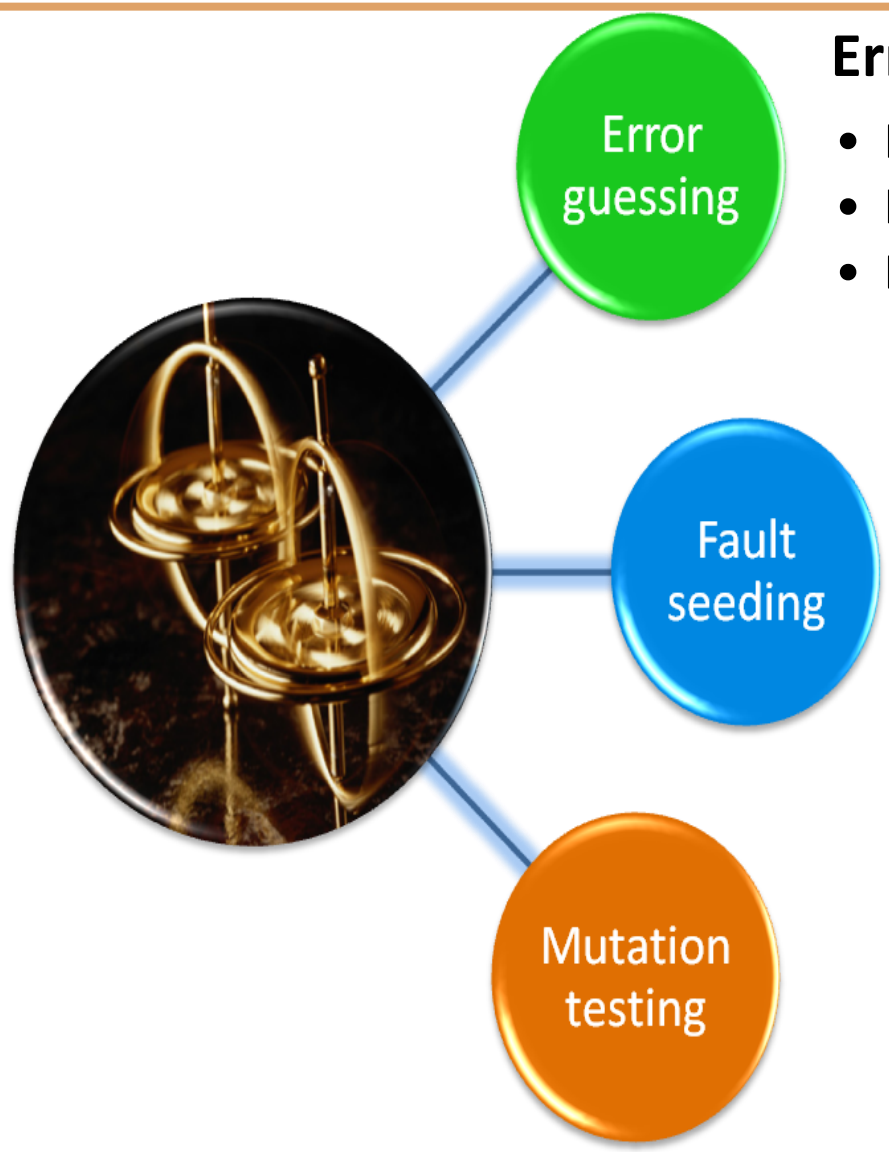
Statement coverage

Executes all statements at least once

No side effects?

Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects.

Testing Types : Testing Technique Based



Error Based

- Most plausible faults
- Historical information
- Experience

Fault Based

- Fault injected into a copy of the code
- Tests run
- Injected and other faults detected is analyzed

Mutation Testing

- Tests seldom-executed code
- Well-defined mutation operators
- Single statement is changed differently in several copies (mutants)
- Tests run against original and mutants
- All mutants should fail in case any of the mutants are not found then the test case is not good enough
- Syntactic faults may reveal more complex, real faults

Testing Types : Testing Technique based approaches

Specification Based

Test the functionality of the system, according to the stated requirements. They test the behavior of the system (system output) given specific input.

Equivalence Partitioning and Boundary Value Analysis are important approaches in specification based because they can uncover entire classes of errors and reduce the total number of test cases that must be developed and executed.

Equivalence partitioning divides input data of a software unit into partitions of data from which test cases can be derived. Test cases are designed to cover each partition.

Boundary value analysis tests are designed to include representatives of boundary values and since boundaries are common locations for errors, that can result in software faults.

Testing Types : Testing Technique based approaches

Intuition Based

Generally rely on the tester's experience to design test cases for the product.

Usage Based

Usage is concerned with finding defects which could be revealed by user as they interact with the product.

Application Domain

Special testing techniques that use specific knowledge about the product domain in order to design test cases. (e.g., GUI, Web-based, real-time, Object-oriented, etc.)

TESTING TYPES

Testing Types : Mode of testing (Manual or Automated)

Manual Testing

- Manual testing involves a human performing the tests step by step, without test scripts.
- Typically used for testing complex tests where automation can be very expensive
- Slow and Tedious
- Hard to get test coverage

E.g. Some of the Acceptance, Black box, White box, Unit tests etc. could be manually tested.

Automated Testing

- Automated testing involves tests which are executed without human assistance, often via test automation frameworks, along with other tools and software
- Needs coding, test framework maintenance but typically is fast and repeatable
- Most efficient for tests which need periodic regular running

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

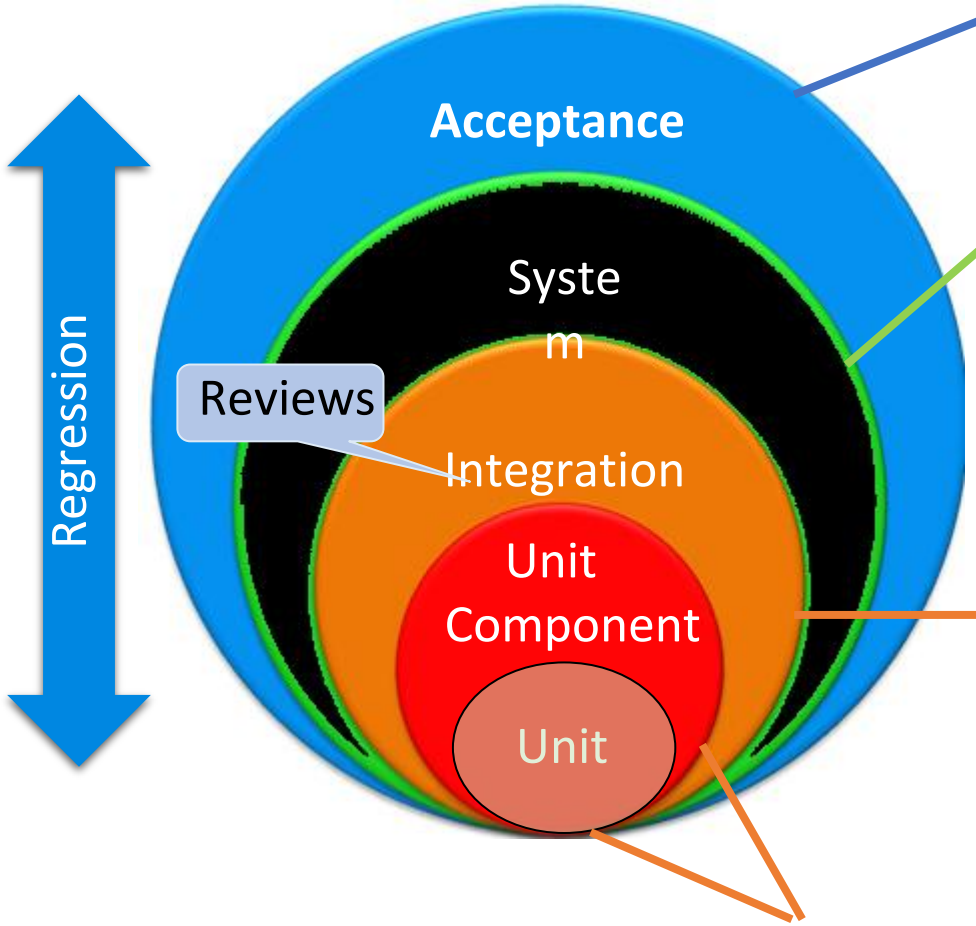
SOFTWARE TESTING

Testing Levels

Dr. Phalachandra H. L
Department of Computer Science and Engineering

TESTING LEVELS

Levels of Testing



- Done by System providers or users or customer
- Determines if system meets needs
- Assessment of whole system behavior
- Collecting information that will help direct the product release
- Discovering bugs that cannot be attributed to individual components

- Focuses on finding interface errors between units or components
- Focus is on the bugs that cannot be seen in unit testing

- Verifies proper functioning of that unit
- No interaction between units

TESTING LEVELS

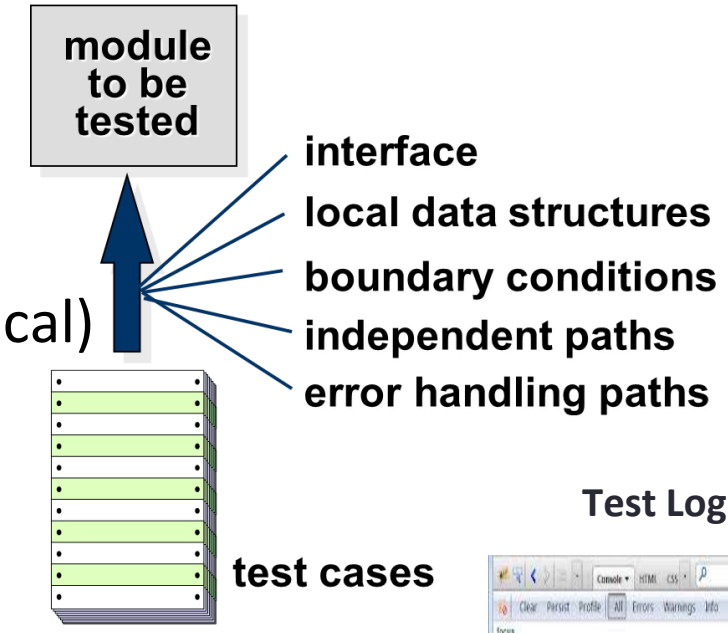
Unit Testing

Focus: Test for coding/construction errors before it goes to QE.

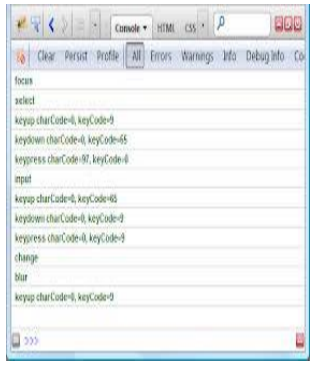
Tests the smallest individually executable code units. Usually done by programmers.

Tests cases could be for

- Algorithms and logic
- Data structures (global and local)
- Interfaces
- Independent paths
- Boundary conditions
- Error handling



Test Log



In OO environment these could be at the class level & minimally constructors/destructors

TESTING LEVELS

Integration Testing

Focus: Test to verify the interfaces between components against a software design.

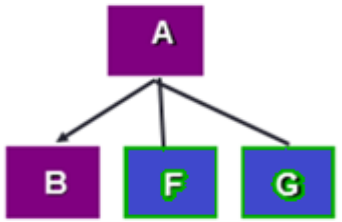
Usually done by programmers & can find issues like timing problems (in real-time systems) & Resource contention problems which are not detectable by unit testing

Approach:

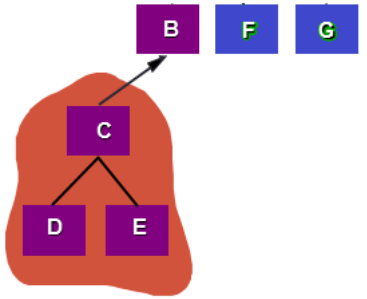
Abstract away unit issues and look for defects between units.

Two strategies ..

- The “big bang” approach - Everything Integrated and then you run to see if the system is functioning as expected or
- Software components corresponding to elements of the architectural design are integrated in an iterative way (since it allows interface issues to be localised mo quickly) either as **top down**



or **bottoms up** and tested until the software works as a system.



TESTING LEVELS

System Testing

- System testing tests a completely integrated system to verify that its compliant with its specified requirements.
- It seeks to detect defects both within the "inter-assemblages" and also within the system as a whole
- Testing in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS).
- Things like the following are tested as part of system testing

TESTING LEVELS

System Testing – Varieties of System Testing

- Smoke and Sanity testing
- Regression testing
  - is used to determine that changes have not caused unintended side effects
- Installation testing
- Functional & Non functional
- Destructive testing
- Software Performance testing
- Usability
- Localization
- Boundary tests
- Startup/shutdown tests
- Platform tests
- Load/stress tests
- Security testing
- Recovery tests
- Cloud Testing
- ..

TESTING LEVELS

Acceptance Testing

- Acceptance testing generally involves running a suite of tests on the completed system.
- Each individual test, known as a case, exercises a particular operating condition of the user's environment or feature of the system, and will result in a pass or fail outcome
- The test environment is usually designed to be identical, or as close as possible, to the anticipated user's environment, including extremes of such
- These tests are created ideally through collaboration between business customers, business analysts, testers, and developers. It's essential that these tests include both

TESTING LEVELS

Acceptance Testing (Cont.)

- These are high-level tests to verify the completeness of a user story or stories 'played' during any sprint/iteration.
- The objective is to provide confidence that the delivered system meets the business requirements of both sponsors and users. The acceptance phase may also act as the final quality gateway, where any quality defects not previously detected may be uncovered. The contractual payment terms are only considered done after the passing of the acceptance test

TESTING LEVELS

Some of these testing could also be categorized as based on the objectives (recap)

Acceptance / qualification testing	Checks the system behavior against the customer's requirements, however these may have been expressed.
Installation testing	Verifies the installation in the target environment. May be identical to system testing in a new environment.
Alpha and beta testing	Before the software is released, it is sometimes given to a small, representative set of potential users for trial use, either in-house (*alpha* testing) or external (*beta* testing). … Alpha and beta use is often uncontrolled, and is not always referred to in a test plan.
Performance testing	Aimed at verifying that the software meets the specified performance requirements (capacity and response time, for instance). A specific kind of performance testing is volume testing, in which internal program or system limitations are tried.

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

Test Planning including Strategy

Dr. Phalachandra H. L
Department of Computer Science and Engineering

TEST PLANNING

Generics

Software test planning is the process of evolving a software test plan which discusses what, when and how testing has to be done as part of the project to ensure quality expectations of the product can be met.

- The out come also serves as a blueprint to conduct software testing activities as a defined process
- Developers, business managers and customers can understand the details of testing
- This can also be used for monitoring and control.
- This software test planning process involves
  1. Ensuring the context and the scope of the project/product under test is well understood
  2. Establishment of the test adequacy criteria

TEST PLANNING

Generics

Software test planning process involves (Contd.)

3. Evolving a test strategy which will be followed for satisfying the objectives for testing
4. Evolving a list of deliverables
5. Creation of detailed test schedule (including estimation of test cases, effort involved for set-up, execution etc.)
6. Planning, Identification and Allocation of resources (people, hardware, software, laboratory, tools, money) needed for testing the project/product
7. Identification of the milestones which will be considered
8. Risk management for the testing activity
9. Establishment of measures & metrics for the project.

TEST PLANNING

1. Understanding and determining the product testing scope

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

- This involves understand the context where the product is going to be used by
  - Reviewing the use case scenario in the product deployment environment
  - Discussing with the designer (Interviews/Meetings)
  - Discussing with the developer
  - Reviewing project/product documentation
  - Play around the product or perform product walk-through
- Given that products and project software components have zillions of combinations its not feasible and practical to exhaustively test all of those different combinations with all of those different techniques for testing.

1. Understanding and determining the product testing scope (Cont.)

- An optimal amount of testing will need to be done, which would mean a subset of the combinations, which would be based on
  - The customer requirement (based on product domain, criticality of the product, its risk assessment and the quality criteria's expected)
  - Project Schedule
  - Project Budget
  - Product Specification
  - Skills & talent of your test team
- At the end of this, there needs to be a clear scope in terms of what would be "in scope" and "out of scope" of the testing.

2. Test Adequacy

- This testing of the subset of possible combinations does not guarantee absence of issues.
- There could also be issues found during testing, all of which may not be feasible to be closed immediately before releasing to customer, due to constraints in terms of
  - No of errors which have been found
  - Schedule not permitting it to be fixed
  - Resources (people, equipment, money) available for fixing the same
  - Some of the issues may be more enhancements
- This leads to **Test Adequacy criteria** which describes the criteria which can be used for determining when to stop testing or consider testing to be complete for that iteration.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

2. Test Adequacy (Cont.)

- This criteria could be a combination of criteria's like
  - All planned tests are completed and post that based on the % of lines of code, or the % of branches, executed .. etc. when the test cases are run

    E.g. - 100% of the statements have been exercised

    - 85% of the branches have been exercised
  - All planned test cases are completed and when there are no Critical high priority issues pending to be fixed
  - It could be when the total number of Severe defects are less than 5 and they are not causing data corruption or stoppage of service

  Etc.

TEST PLANNING

3. Testing Strategy

Strategy is also known as test approach and defines how testing would be carried out and deals with the following to achieve the testing objectives

1. The testing mindset or the model which will be followed
2. What test types will be used as part of the process
3. Test environment which will be used
4. Automation strategy
5. Tools
6. Risk analysis with contingency planning for the strategy

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

It aims to align different stakeholders of quality assurance in terms of terminology, test and integration levels etc.

3.1 Testing Models or Mindsets

Demonstration Model/Mindset

- To make sure that the software runs and solves the problem
- If the software passes all tests from the test set then it would establish that it satisfies the specs
- Given the developers and testers are aware, unconsciously they might end up testing what would succeed
- Not advocated easily

Evaluation Model/Mindset

- Detects faults through the lifecycle phases
- Focus on analysis and review techniques to detect faults in requirements and design documents

TEST PLANNING : Testing Strategy

3.1 Testing Models or Mindsets

Destruction Model/Mindset

- Try to make the software fail and find as many faults
- Good and effective test cases are those that find faults
- Difficult to decide when to stop testing as we do not know the number of faults left in the system

Preventive Model/Mindset

- Prevents faults in early phases through careful planning and design of test activities
- Reviews and test driven development falls into this category

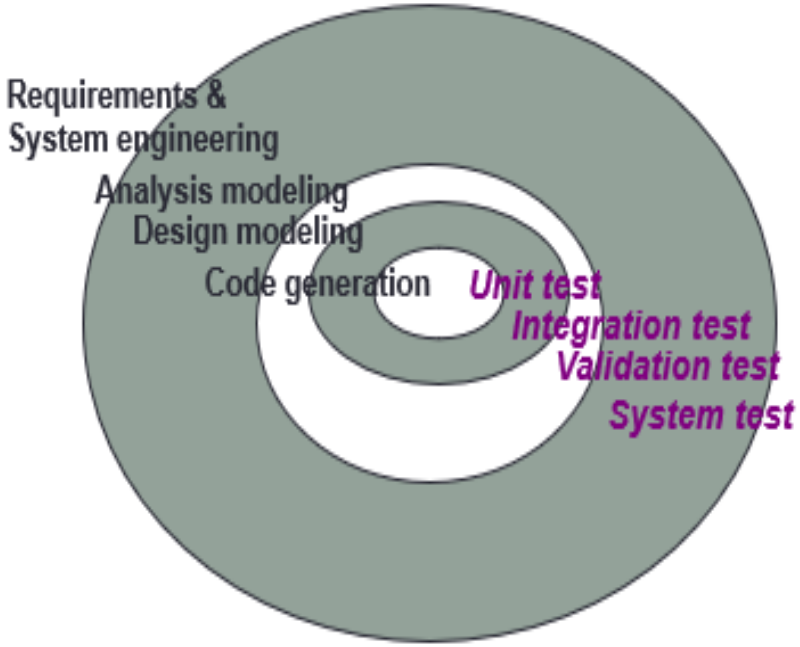
3.2 Testing Types Chosen

This involves what are the different testing types which would be used as part of the strategy to test the project.

- **Each of the lifecycle phases has outcomes which can be tested (statically or dynamically)**
  - Feasibility phase has the acceptance test cases
  - Requirements phase has the requirement specification and thus may have functional and system test cases based on the requirements which can be reviewed
  - Architecture/Design phase has the refined functional and system test cases and integration test cases which can be reviewed
  - Implementation phase has code unit test cases which can be reviewed
  - Testing phases would involve reviews and execution of all test cases designed
  - Maintenance phase would involve review and execution of regression and other tests

3.2 Testing Types Chosen (Cont.)

- **There were a number of different types of testing which have been discussed in the last couple of classes.**
  - Some of these would be chosen as part of this step
- **This may involve strategies in terms of**
  - Begin by 'testing-in-the-small' and move toward 'testing-in-the-large'
  - Top Down and Bottoms Up Testing
  - Positive and Negative Testing
  - Functional and Non Functional testing
  - Dynamic and Heuristics based approach



3.3. Test Execution Environment which will be used

- A **testing environment** or **test execution environment** or a **test bed** is a setup of software and hardware for the testing teams to execute test cases.
- Test bed is configured as per the need of the Application Under Test
- Setting up a right test environment ensures success of software testing else it could result in delay, cost escalations and incorrect conclusions
- Components of the test execution environment could include **system or application under test**, ***test data***, ***DB***, ***front end*** for the test environment, ***OS on the Server***, ***Servers***, ***Storage*** and ***Network*** and all documents needed for these hardware and software infrastructure.
- Test environment Management involves maintenance and upkeep of the test bed and may involve monitoring and modifying components based on requirements, performance etc.
- Challenges towards setting up a test environment could be in terms of proper planning on resource usage, remote environment, setup time, sharing of the environment and setting up complex configurations

TEST PLANNING : Testing Strategy

3.4 Automation Strategy

- Defining Goals
- Planning the test approach
- Selection of Automation framework
- Selecting test tool
- Test case design and execution
- Maintaining script

TEST PLANNING : Testing Strategy

3.5 Tools which will be used

- The technology of the Software or application under test (AUT) will need to be compatible to the tool and drives the selection of it.
- Testers will need to be comfortable with the tool, else it may not be effectively used
- Tools chosen needs to be balanced in terms of the features offered, ability to generate reports/data needed for different stakeholders and the ease of use.
- Cross platform support is an expectation as automated tests would/may need to run on different platforms.
- Acceptability/Popularity/prevalence of the tool in the Industry is an indication of availability of support, quality documentation, technical forums and availability of trained personnel.
- Cost
- Opensource or Proprietary have different characteristics of cost, features and the support and needs to be balanced

TEST PLANNING : Testing Strategy

3.6 Risk Analysis with Contingency Planning

Risk is the probability of an unwanted incident during or towards testing

Risks in strategy could be in terms of

- Changes to the Business, Technology or competition directions
- Resources
- Quality of the software product being developed
- The test models not being able to be used
- Some type of testing chosen cannot be use
- Test environment and its state
- Automation or
- Tool issues

Any risks found in any of them would need to be planned for addressing as part of the mitigation and contingency.

4. Evolving a list of deliverable

- In this step the list of activities and deliverables will be identified which would need to executed and delivered.
- This could include test specifications for each of the modules of the product, test cases for different conditions planned.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

5. Creation of detailed test schedule

- This including estimation for building test strategy/specification/test cases/test environment setup and test execution, test reporting etc.)
- This would include WBS and estimation using some of the techniques used earlier (similar to project planning).

This would lead to something like

Create test scenarios and test cases - 100 man hours

Execute test cases - 200 man hours

.....

- We would associate the same into a calendar

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

6. Planning, Identification and Allocation of resources

- This and the previous step of estimating and scheduling are done together or iterated to ensure that the schedule factors in the characteristics of the planned resources.
- Initially this will involve
  - Identifying the number and type of servers, storage, test tools and the network resources needed for the activities identified
  - Identifying the number and type of people to work on the project. This could be in the roles of test manager, testers, test developers, test administrators and SQA with the requisite skills and skill level.
  - Identifying the test environment (test rings etc.) which would be set up with the resources identified along with identification of the test data sets etc.

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

7 & 8 : Identification of the milestones & Risks

- Once the resources are identified and their capacities, skills are identified, the schedule is reworked.
- Considering the deliverables expected, the schedule and commitments if any to the customers, the project milestones are identified.
- Risks for completion of each of the task from a schedule and quality perspective is identified, analyzed, mitigation plans made and the triggers for kick-off identified.
- These milestones are used to track or monitor progress and control overruns. These milestones are also used to identify any risk triggers and to help kick-in the mitigation plans.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

9 : Measures are identified and Metrics identified

- The measurements which will be made like the number of test cases planned and created, number of test cases run, the amount of time spent on creation, execution, the number of errors found etc.
- The errors are classified as per define policies on whether they are critical, serious, medium or low impact.
- Metrics like the number of test cases executed/day or % of test cases executed to planned, number of issues/KLoC, number of critical issues/KLoC are planned for measurement and evaluation, No of requirements traced across the lifecycle etc.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

Test Planning including Strategy

Dr. Phalachandra H. L
Department of Computer Science and Engineering

TEST PLANNING

Generics

Software test planning is the process of evolving a software test plan which discusses what, when and how testing has to be done as part of the project to ensure quality expectations of the product can be met.

- The out come also serves as a blueprint to conduct software testing activities as a defined process
- Developers, business managers and customers can understand the details of testing
- This can also be used for monitoring and control.
- This software test planning process involves
  1. Ensuring the context and the scope of the project/product under test is well understood
  2. Establishment of the test adequacy criteria

TEST PLANNING

Generics

Software test planning process involves (Contd.)

3. Evolving a test strategy which will be followed for satisfying the objectives for testing
4. Evolving a list of deliverables
5. Creation of detailed test schedule (including estimation of test cases, effort involved for set-up, execution etc.)
6. Planning, Identification and Allocation of resources (people, hardware, software, laboratory, tools, money) needed for testing the project/product
7. Identification of the milestones which will be considered
8. Risk management for the testing activity
9. Establishment of measures & metrics for the project.

TEST PLANNING

1. Understanding and determining the product testing scope

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

- This involves understand the context where the product is going to be used by
  - Reviewing the use case scenario in the product deployment environment
  - Discussing with the designer (Interviews/Meetings)
  - Discussing with the developer
  - Reviewing project/product documentation
  - Play around the product or perform product walk-through
- Given that products and project software components have zillions of combinations its not feasible and practical to exhaustively test all of those different combinations with all of those different techniques for testing.

1. Understanding and determining the product testing scope (Cont.)

- An optimal amount of testing will need to be done, which would mean a subset of the combinations, which would be based on
  - The customer requirement (based on product domain, criticality of the product, its risk assessment and the quality criteria's expected)
  - Project Schedule
  - Project Budget
  - Product Specification
  - Skills & talent of your test team
- At the end of this, there needs to be a clear scope in terms of what would be "in scope" and "out of scope" of the testing.

2. Test Adequacy

- This testing of the subset of possible combinations does not guarantee absence of issues.
- There could also be issues found during testing, all of which may not be feasible to be closed immediately before releasing to customer, due to constraints in terms of
  - No of errors which have been found
  - Schedule not permitting it to be fixed
  - Resources (people, equipment, money) available for fixing the same
  - Some of the issues may be more enhancements
- This leads to **Test Adequacy criteria** which describes the criteria which can be used for determining when to stop testing or consider testing to be complete for that iteration.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

2. Test Adequacy (Cont.)

- This criteria could be a combination of criteria's like
  - All planned tests are completed and post that based on the % of lines of code, or the % of branches, executed .. etc. when the test cases are run

    E.g. - 100% of the statements have been exercised

    - 85% of the branches have been exercised
  - All planned test cases are completed and when there are no Critical high priority issues pending to be fixed
  - It could be when the total number of Severe defects are less than 5 and they are not causing data corruption or stoppage of service

  Etc.

TEST PLANNING

3. Testing Strategy

Strategy is also known as test approach and defines how testing would be carried out and deals with the following to achieve the testing objectives

1. The testing mindset or the model which will be followed
2. What test types will be used as part of the process
3. Test environment which will be used
4. Automation strategy
5. Tools
6. Risk analysis with contingency planning for the strategy

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

It aims to align different stakeholders of quality assurance in terms of terminology, test and integration levels etc.

3.1 Testing Models or Mindsets

Demonstration Model/Mindset

- To make sure that the software runs and solves the problem
- If the software passes all tests from the test set then it would establish that it satisfies the specs
- Given the developers and testers are aware, unconsciously they might end up testing what would succeed
- Not advocated easily

Evaluation Model/Mindset

- Detects faults through the lifecycle phases
- Focus on analysis and review techniques to detect faults in requirements and design documents

TEST PLANNING : Testing Strategy

3.1 Testing Models or Mindsets

Destruction Model/Mindset

- Try to make the software fail and find as many faults
- Good and effective test cases are those that find faults
- Difficult to decide when to stop testing as we do not know the number of faults left in the system

Preventive Model/Mindset

- Prevents faults in early phases through careful planning and design of test activities
- Reviews and test driven development falls into this category

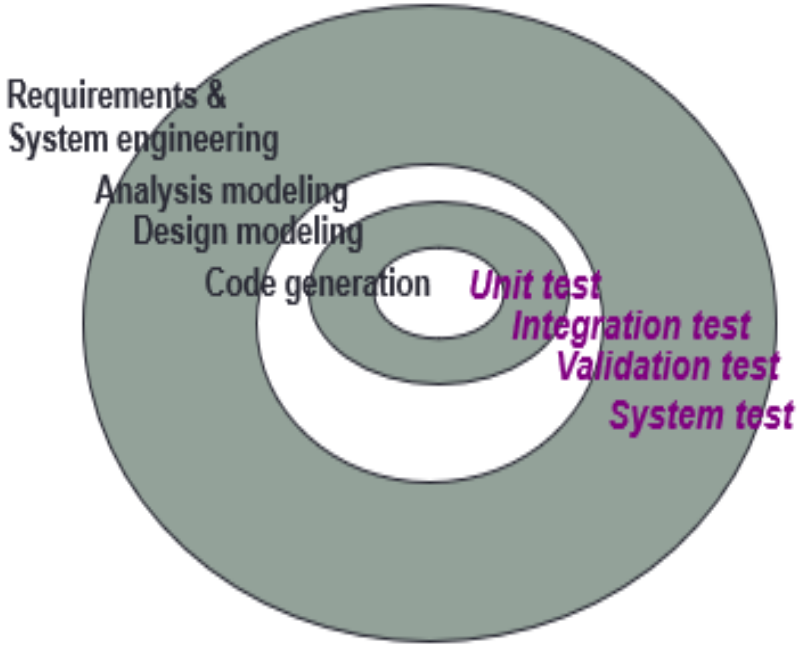
3.2 Testing Types Chosen

This involves what are the different testing types which would be used as part of the strategy to test the project.

- **Each of the lifecycle phases has outcomes which can be tested (statically or dynamically)**
  - Feasibility phase has the acceptance test cases
  - Requirements phase has the requirement specification and thus may have functional and system test cases based on the requirements which can be reviewed
  - Architecture/Design phase has the refined functional and system test cases and integration test cases which can be reviewed
  - Implementation phase has code unit test cases which can be reviewed
  - Testing phases would involve reviews and execution of all test cases designed
  - Maintenance phase would involve review and execution of regression and other tests

3.2 Testing Types Chosen (Cont.)

- **There were a number of different types of testing which have been discussed in the last couple of classes.**
  - Some of these would be chosen as part of this step
- **This may involve strategies in terms of**
  - Begin by 'testing-in-the-small' and move toward 'testing-in-the-large'
  - Top Down and Bottoms Up Testing
  - Positive and Negative Testing
  - Functional and Non Functional testing
  - Dynamic and Heuristics based approach



3.3. Test Execution Environment which will be used

- A **testing environment** or **test execution environment** or a **test bed** is a setup of software and hardware for the testing teams to execute test cases.
- Test bed is configured as per the need of the Application Under Test
- Setting up a right test environment ensures success of software testing else it could result in delay, cost escalations and incorrect conclusions
- Components of the test execution environment could include **system or application under test**, ***test data***, ***DB***, ***front end*** for the test environment, ***OS on the Server***, ***Servers***, ***Storage*** and ***Network*** and all documents needed for these hardware and software infrastructure.
- Test environment Management involves maintenance and upkeep of the test bed and may involve monitoring and modifying components based on requirements, performance etc.
- Challenges towards setting up a test environment could be in terms of proper planning on resource usage, remote environment, setup time, sharing of the environment and setting up complex configurations

TEST PLANNING : Testing Strategy

3.4 Automation Strategy

- Defining Goals
- Planning the test approach
- Selection of Automation framework
- Selecting test tool
- Test case design and execution
- Maintaining script

TEST PLANNING : Testing Strategy

3.5 Tools which will be used

- The technology of the Software or application under test (AUT) will need to be compatible to the tool and drives the selection of it.
- Testers will need to be comfortable with the tool, else it may not be effectively used
- Tools chosen needs to be balanced in terms of the features offered, ability to generate reports/data needed for different stakeholders and the ease of use.
- Cross platform support is an expectation as automated tests would/may need to run on different platforms.
- Acceptability/Popularity/prevalence of the tool in the Industry is an indication of availability of support, quality documentation, technical forums and availability of trained personnel.
- Cost
- Opensource or Proprietary have different characteristics of cost, features and the support and needs to be balanced

TEST PLANNING : Testing Strategy

3.6 Risk Analysis with Contingency Planning

Risk is the probability of an unwanted incident during or towards testing

Risks in strategy could be in terms of

- Changes to the Business, Technology or competition directions
- Resources
- Quality of the software product being developed
- The test models not being able to be used
- Some type of testing chosen cannot be use
- Test environment and its state
- Automation or
- Tool issues

Any risks found in any of them would need to be planned for addressing as part of the mitigation and contingency.

4. Evolving a list of deliverable

- In this step the list of activities and deliverables will be identified which would need to executed and delivered.
- This could include test specifications for each of the modules of the product, test cases for different conditions planned.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

5. Creation of detailed test schedule

- This including estimation for building test strategy/specification/test cases/test environment setup and test execution, test reporting etc.)
- This would include WBS and estimation using some of the techniques used earlier (similar to project planning).

This would lead to something like

Create test scenarios and test cases - 100 man hours

Execute test cases - 200 man hours

.....

- We would associate the same into a calendar

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

6. Planning, Identification and Allocation of resources

- This and the previous step of estimating and scheduling are done together or iterated to ensure that the schedule factors in the characteristics of the planned resources.
- Initially this will involve
  - Identifying the number and type of servers, storage, test tools and the network resources needed for the activities identified
  - Identifying the number and type of people to work on the project. This could be in the roles of test manager, testers, test developers, test administrators and SQA with the requisite skills and skill level.
  - Identifying the test environment (test rings etc.) which would be set up with the resources identified along with identification of the test data sets etc.

Test Planning
1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

7 & 8 : Identification of the milestones & Risks

- Once the resources are identified and their capacities, skills are identified, the schedule is reworked.
- Considering the deliverables expected, the schedule and commitments if any to the customers, the project milestones are identified.
- Risks for completion of each of the task from a schedule and quality perspective is identified, analyzed, mitigation plans made and the triggers for kick-off identified.
- These milestones are used to track or monitor progress and control overruns. These milestones are also used to identify any risk triggers and to help kick-in the mitigation plans.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

TEST PLANNING

9 : Measures are identified and Metrics identified

- The measurements which will be made like the number of test cases planned and created, number of test cases run, the amount of time spent on creation, execution, the number of errors found etc.
- The errors are classified as per define policies on whether they are critical, serious, medium or low impact.
- Metrics like the number of test cases executed/day or % of test cases executed to planned, number of issues/KLoC, number of critical issues/KLoC are planned for measurement and evaluation, No of requirements traced across the lifecycle etc.

Test Planning

1. Context/Scope
2. Test Adequacy Criteria
3. Test Strategy
4. List of Deliverables
5. Creation of test schedule
6. Planning, Identification and allocation of resources
7. Milestones
8. Risk Management
9. Establish measures and metrics

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

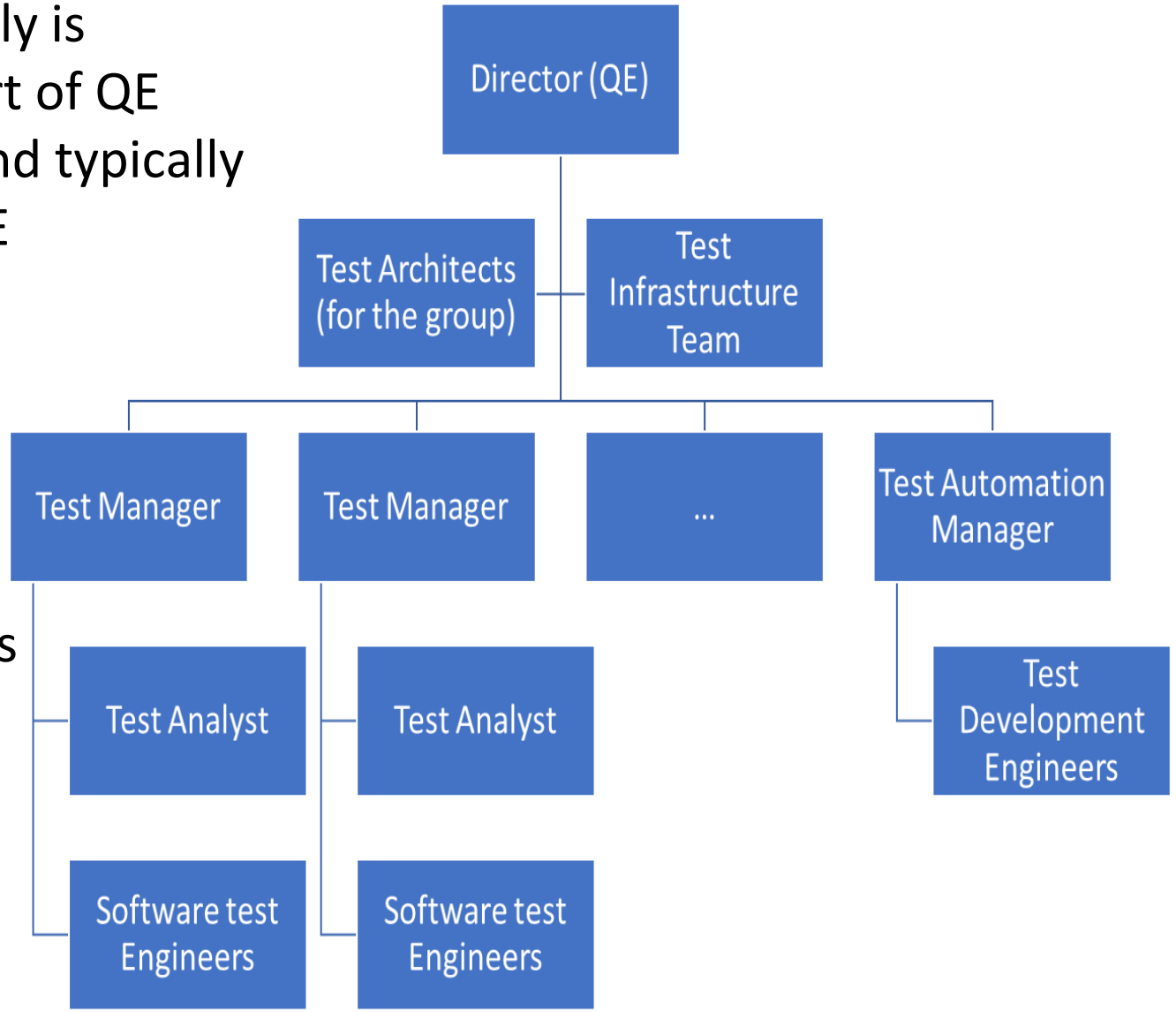
Test Roles and Testing Lifecycle and Process

Dr. Phalachandra H. L
Department of Computer Science and Engineering

TEST ROLES

Structure of Testing Organization

- Software testing typically is considered as being part of QE (Quality Engineering) and typically is executed from the QE organization
- Given that the roles, responsibilities and the skills for the QE activity is different from the other groups, QE groups are typically organized as shown (some of the roles may/may not exist and they can be called differently too)



TEST ROLES
Roles and Responsibilities

Role	Responsibilities
Test Director	Provide oversight, Co-ordination, Provide Strategic vision, High level Customer and stake holder connect
Test Manager	Prepare test strategy, plans for the project (or product) and monitor and control the testing process
Test Infrastructure Manager	Manages all of the infrastructure for the testing teams, capacity planning, maintenance, support, configurations
Test Automation Manager	Manages the development of tools when planned for and scripts for automating the tests as per plans.
Test Architect	Designs test infrastructure, helps picking the appropriate tools or helps driving the requirements for tools, supports, validates the test strategy being planned as part of test planning
Test Analyst	Supports mapping of customer environment and test conditions and features to testing conditions and documentation
Software Test Engineer	Tests the product or project system outcomes using the appropriate planned testing techniques and tools
Test Development Engineer	Develops tools and scripts for automating the tests planned

TESTING PROCESS

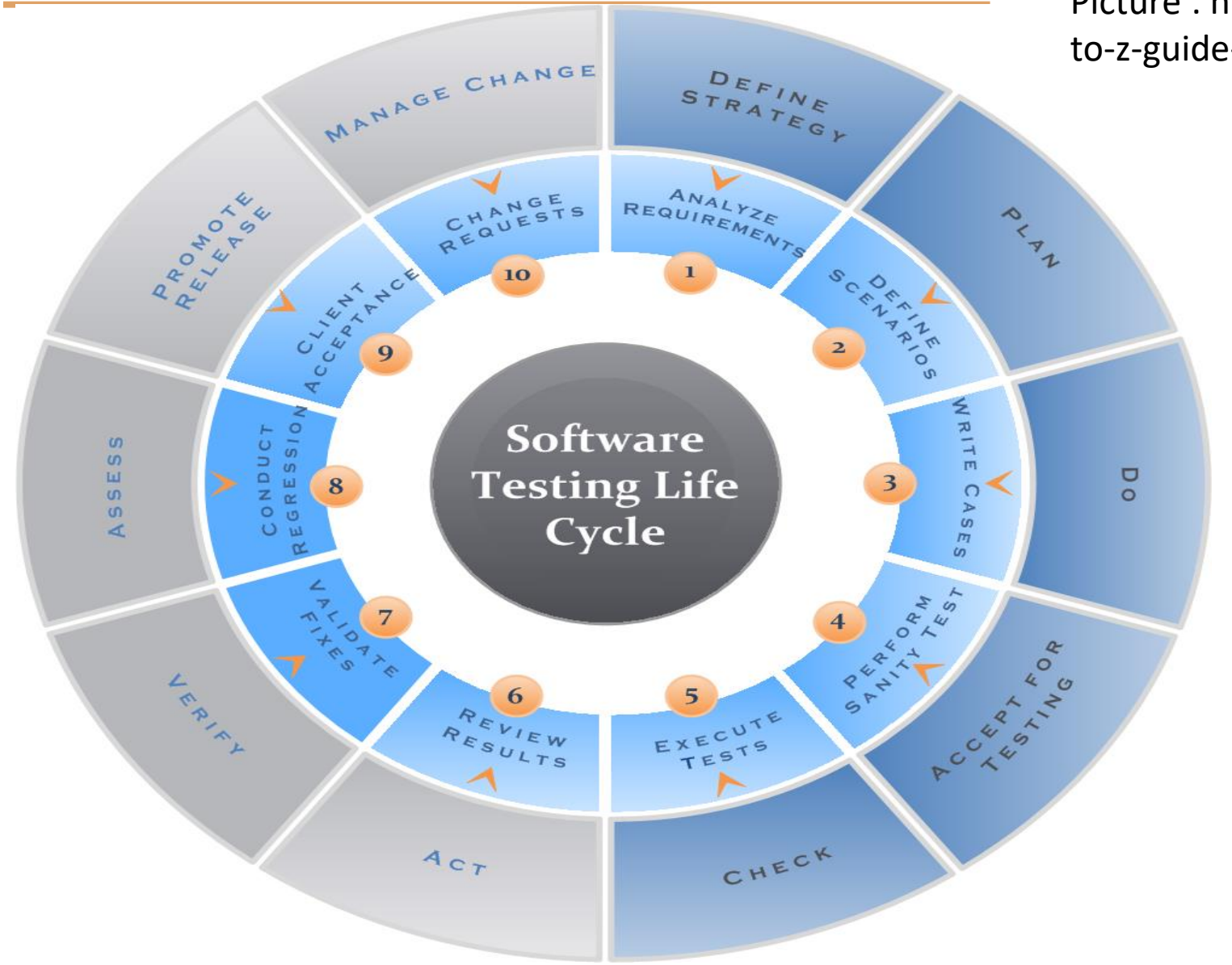
Test Process

- Testing is a process and goes through a Lifecycle with different steps
- The activities of testing would have the following steps

Planning and Control	▪ Involves creating the Test Strategy ▪ Creating Test Sufficiency criteria ▪ Planning for resources and building a Schedule ▪ Setup review points, status reporting mechanisms, approval boards ..
Analysis and Design	▪ Analyze test requirements, product architecture and interfaces ▪ Identify test conditions and design tests and test environment
Implementation and Execution	▪ Develop test cases, test data, test automation ▪ Execute tests, collect metrics, log results & compare with expected values.
Evaluate exit criteria & Reporting	▪ Setting up the test completion/stopping criteria based on the application functionality and objectives E.g. Test cases, Pass percentage, Bug rate, Deadlines, RTM
Test Closure activities	▪ Done when the testing is complete or project is cancelled ▪ Verify all planned deliverables are completed, issues resolved ▪ Archive all test scripts, environment and formally close with reports and do a retrospective

TESTING PROCESS

Software test life cycle



Picture : https://reqtest.com/testing-blog/the-a-to-z-guide-to-the-software-testing-process/

TESTING PROCESS

Software test Execution

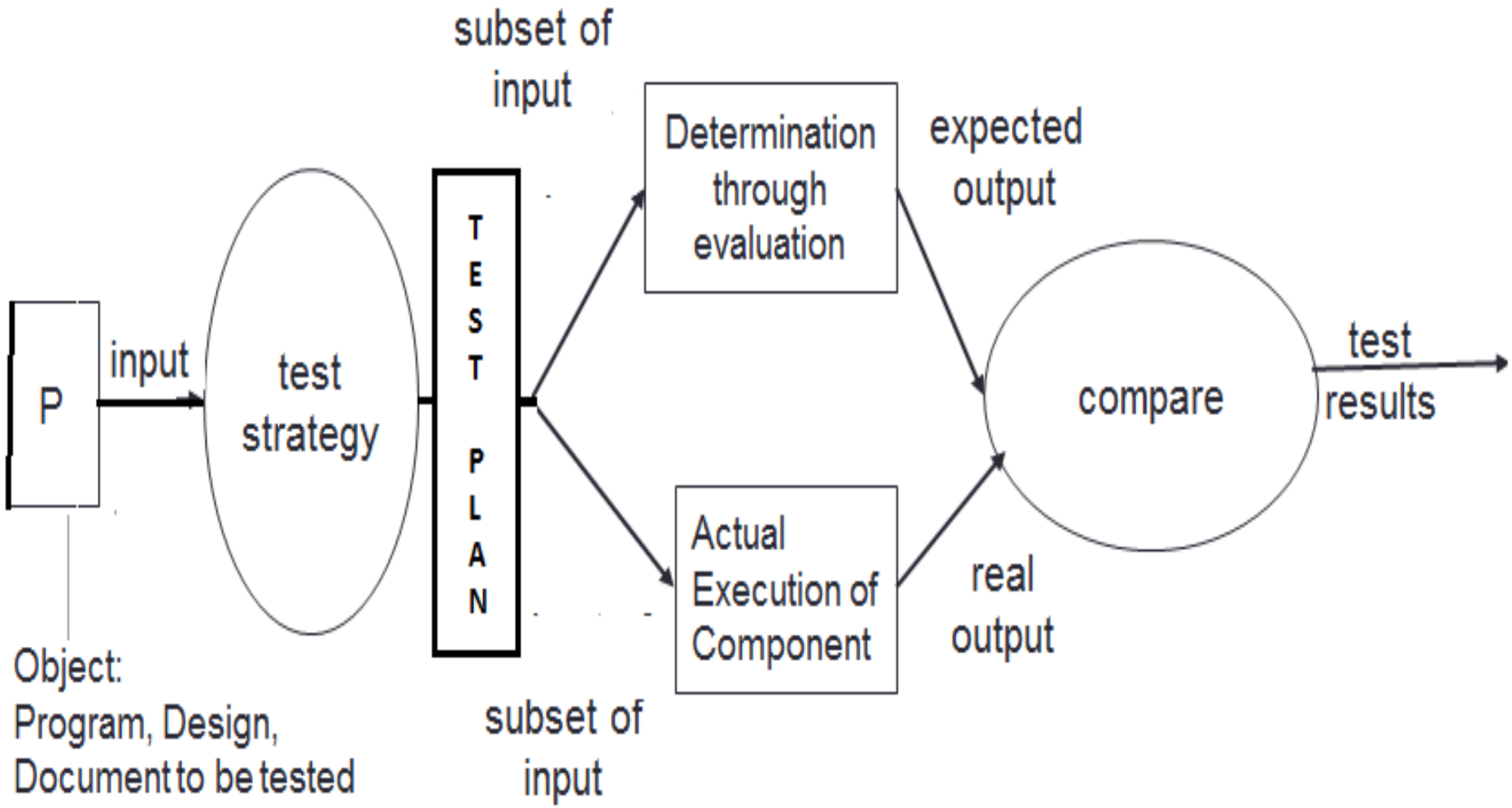
It’s the process of executing the code and comparing the actual observations of results to the predicted expected values.

The following are the actions typically taken for test execution:

- Based on the context a subset of test cases is selected for execution in that test this cycle.
- These test cases are then assigned to testers for execution.
- Environment is setup-configured-test data is setup-steps for execution is noted, expected output looked at
- Tests are executed, results logged, status captured and bugs logged.
- In case testing is blocked, they are worked around or resolved and then continued. Bugs/issues are triaged and passed on to respective teams
- Results reported, measurements done, metrics & test results analyzed

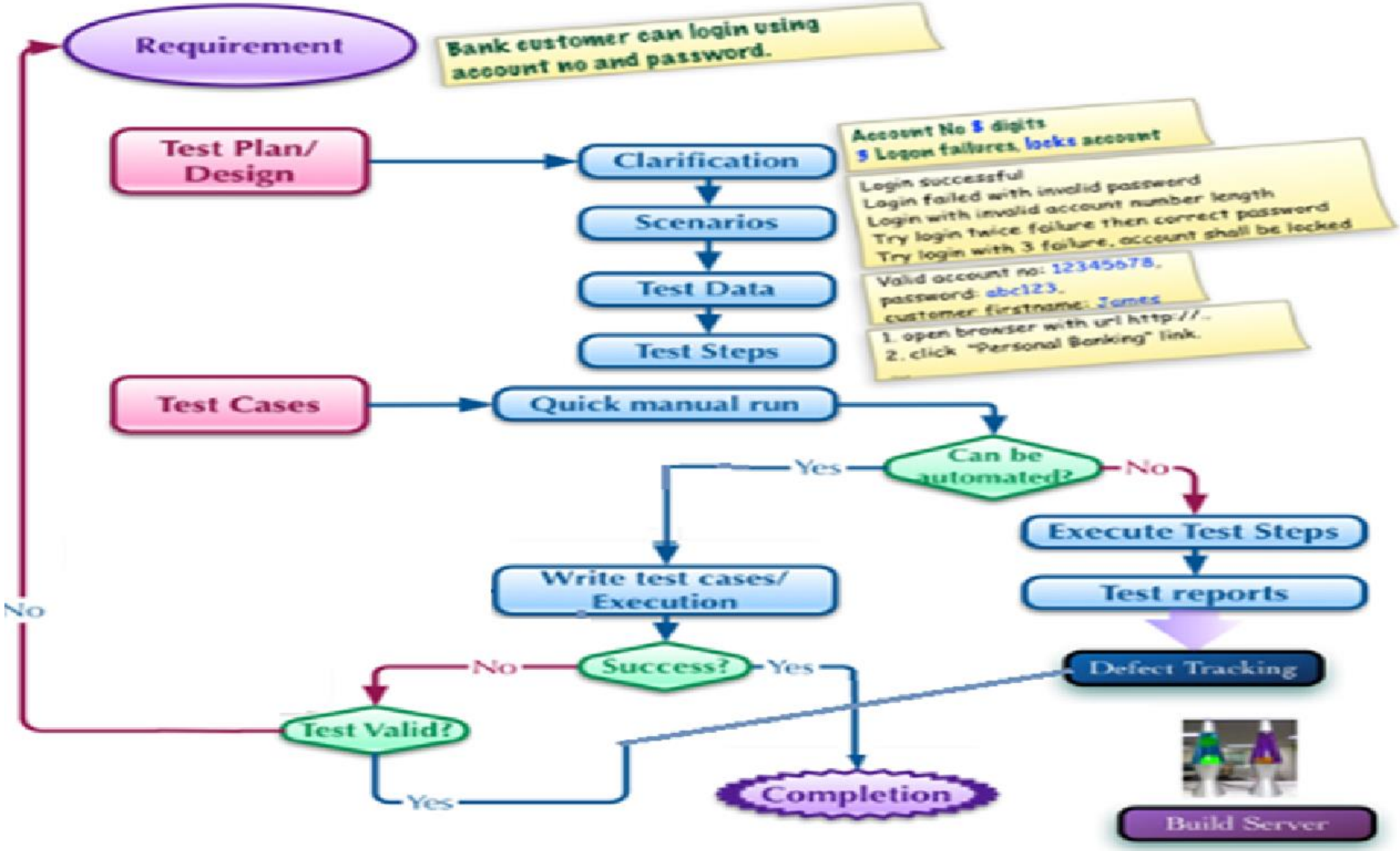
TESTING PROCESS

Test Execution Process



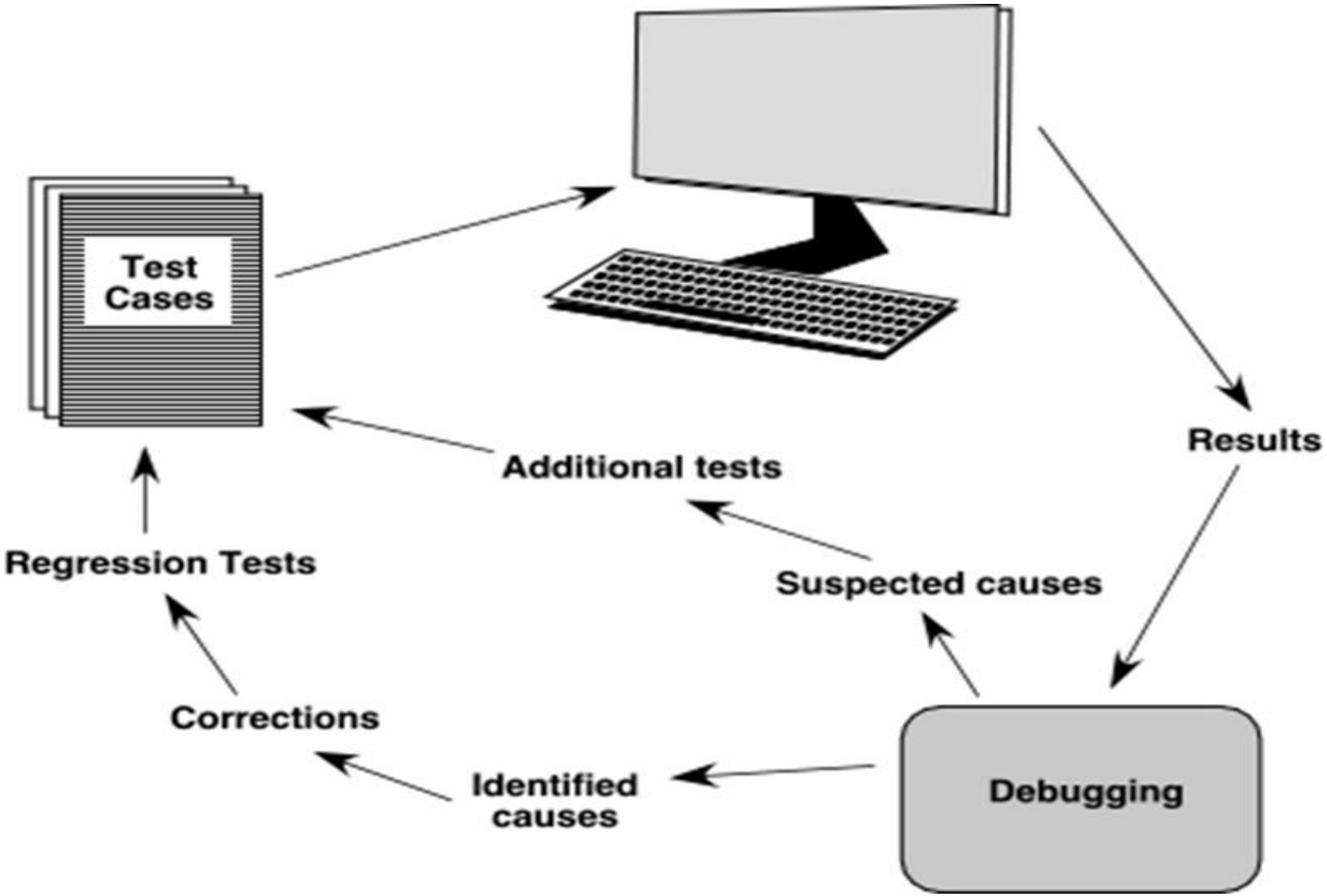
TESTING PROCESS

Test Execution Process (an example)



TESTING PROCESS

Process for debugging an issue during testing



THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4 - SOFTWARE TESTING

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE TESTING

Test Measures, Metrics, Reliability, Practices and Tools

Dr. Phalachandra H. L
Department of Computer Science and Engineering

SOFTWARE TESTING

Software Test Related Measurements and Metrics

Software Test Metric is a quantitative measure of the testing process indicating the progress, quality, productivity and the degree to which a system, system components possesses a given attribute.

The goal of software testing metrics is to

- Improve the efficiency and effectiveness in the software testing process
- To help make better decisions for future testing processes by providing reliable data about the testing process

Metrics are supported by **measurements** is critical in performing SQA and testing. Measurements support increasing the effectiveness of the testing processes

- When used as part of the project planning and monitoring for the optimizing planning, executing and monitoring of the test activities
- by providing data which is used to evaluate the quality of the product under test.
- By helping in defect analysis, which can help in improving the quality of future projects

These measures can

- Represent the quality of the product
- Represent test coverage & hence the thoroughness of the test set.

Software Test Related Metric Characteristics

1.Quantitative:

Metrics must possess quantitative nature. It means metrics can be expressed in values.

2.Understandable:

Metric computation should be easily understood ,the method of computing metric should be clearly defined.

3.Applicability:

Metrics should be applicable in the initial phases of development of the software.

4.Repeatable:

The metric values should be same when measured repeatedly and consistent in nature.

5.Economical:

Computation of metric should be economical.

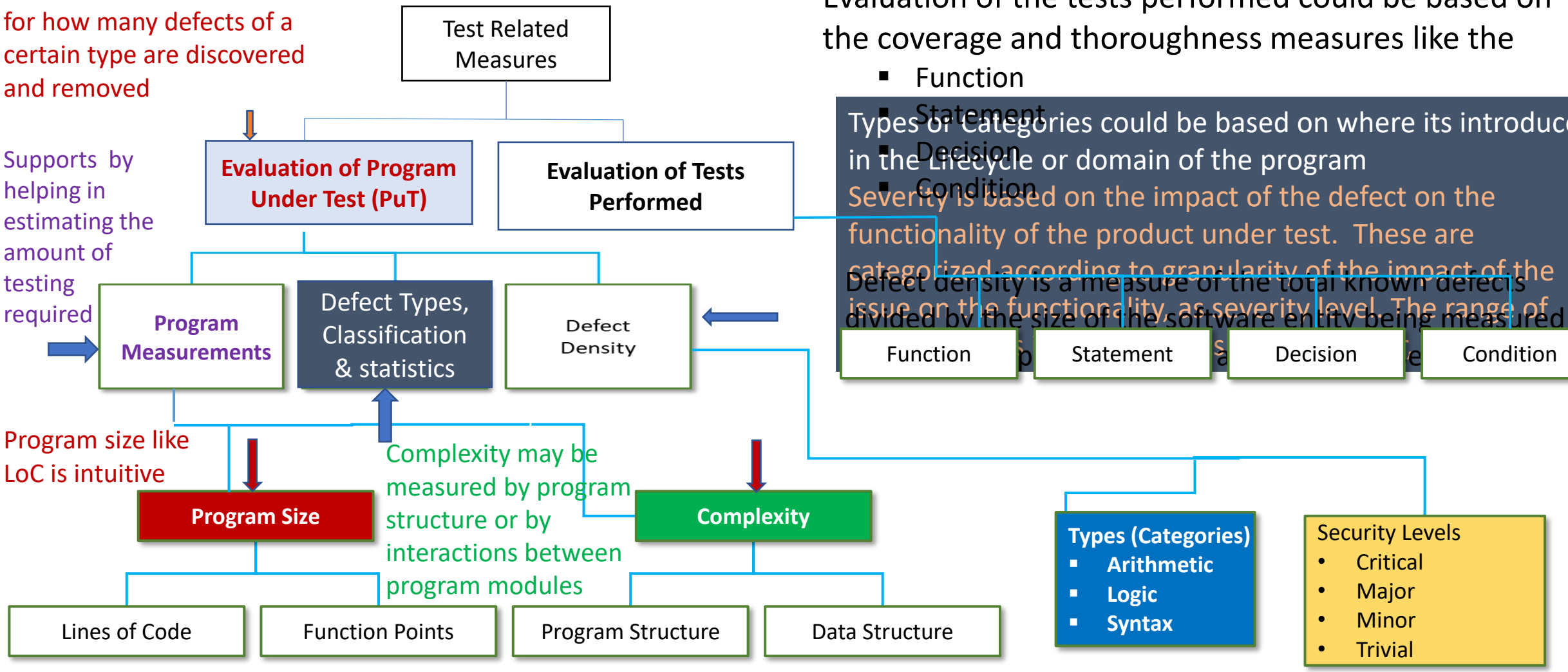
6.Language Independent:

Metrics should not depend on any programming language.

SOFTWARE TESTING

Software Test Measurements

Test related measures can be looked at as follows



Evaluation of the tests performed could be based on the coverage and thoroughness measures like the

- Function
- Statement
- Decision
- Condition

Types or Categories could be based on where its introduced in the Lifecycle or domain of the program

Severity is based on the impact of the defect on the functionality of the product under test. These are categorized according to granularity of the impact of the issue on the functionality, as severity level. The range of

Defect density is a measure of the total known defects divided by the size of the software entity being measured

Function | Statement | Decision | Condition

Software Test Measurements

These test-related measures could also be looked at from the following perspectives

- Product measures (e.g. size, complexity)
- Process measures targeted to improve the efficiency of processes (e.g. number of test cases designed/phase)
- Project progress related measures (e.g. Time spent/number of test cases developed)

Software Test Related Measures – Product Related

Metrics	Description
SLOC	Size in lines of code
Fault Density	The ratio of # of faults found to the size of the programs(bugs/LOC). Higher density represents lower quality
MTBF	Mean time between failure measured in hours. Based on statistical analysis that indicate the probability of failure
Failure Rate	Inverse of MTBF
Defect Distribution	% of defects attributed to a specified phases in SDLC
Defect Density of modules	The ration of # of faults found in the module to the total faults found in the product
Defect Leakage	Indicates test efficiency = (Total Number of Defects Found in UAT/ Total Number of Defects Found Before UAT) x 100

SOFTWARE TESTING

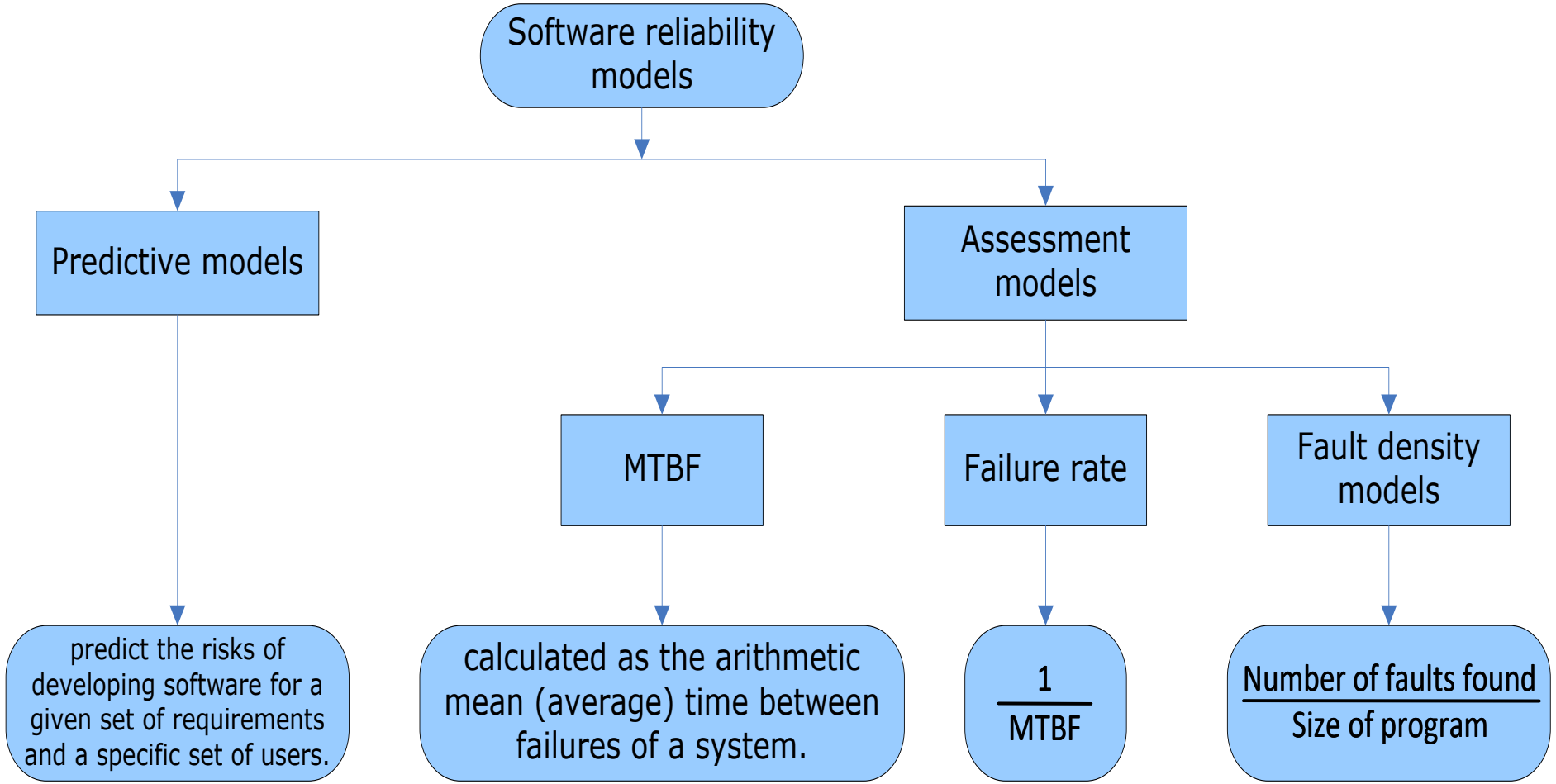
Software Test Coverage Metrics – Product Related

Metrics	Description
Percent Test Coverage	Includes: Statement Coverage, Branch/Decision Coverage, Condition Coverage, Loop Coverage, Path Coverage, Data Flow Coverage, etc.
Percent completed and Percent Defects corrected	# of test cases executed and closed compared to total # of test cases & # of defects fixed/closed compared to total # of defects
Defect Removal Effectiveness	% of defects found at the later phase of development.
Requirement Compliance Factor	Using the traceability matrix, the RCF measures the coverage provided by a test case to one or set of requirements.
Defect Discovery Rate	Number of defects found per line of code (LOC).
Defect Removal Cost	The cost associated with finding and fixing a defect.
Cost of Quality	Total cost of prevention, appraisal, rework/failure, to the total cost of the project.
Percent Injected Fault Discovered	Number of defects injected found, compared to total number of faults injected.
Mutation Score	Number of killed mutants to the total number of mutants. Again, any mutant left alive points to the weakness in the testing effort.

SOFTWARE TESTING

Software Reliability

Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment



Prevention Model

First write the tests, *then* do the design/implementation

Could be done as part of some of the agile approaches like XP

Supported by tools, e.g. JUnit

Is more than a mere test technique; it subsumes part of the design work

Steps associated with TDD

1. Add a test
2. Run this and the earlier tests, and see that the system fails
3. Make a small change add something small to make the test work
4. Continue incrementally running all the planned tests till you can see they all run properly
5. Refactor the system to improve its design and remove redundancies

Selenium automation suite has four main offerings:

- Selenium IDE
- Selenium RC
- Selenium WebDriver
- Selenium Grid

Why Selenium?

- Open source
- Highly extensible
- Can run tests across different browsers
- Supports various operating systems
- Supports mobile devices
- Can execute tests in parallel

Short Comings of Selenium

- Can only test web applications
- Has no built-in object repository
- Automates at a slower rate because it does not have a native IDE and only third party IDE can be used for development
- Data-driven testing is more cumbersome since you have to rely on the programming language's capabilities for setting values for your test data
- Cannot access elements outside the web application under test
- No official user support is being offered.

Tools - Junit

- JUnit is a simple framework to write repeatable tests.
- JUnit is a unit testing framework for Java programming language.
- It eliminates the need to manually verify and tally results and supports faster product deployment.
- JUnit test framework provides the following important features –
  - Fixtures
  - Test suites
  - Test runners
  - JUnit classes

Some Annotations used in Junit

@Before Used to execute some statement such as a precondition before each test case

@BeforeClass Used to execute some statements before all the test cases.

@After Used to execute some statements after each test case ; e.g. resetting variables, deleting variables.

@AfterClass Used to execute some statements after all the testcases ; e.g. releasing resources.

@Ignore Used to ignore some statements during test execution; e.g. to ignore a particular test case during test execution.

Tools - Junit

Advantages of Junit

- Alternate front ends to display results of tests are available
- Separate class loaders for each unit test to avoid side effects
- Provides methods like setUp and tearDown for standard resource initialization
- Set of assert methods to check results of tests
- Integration with popular tools such as Ant and Maven and IDEs like Eclipse and Jbuilder

Shortcomings of Junit

- Cannot do, dependency testing
- Not suitable for high level testing
- Large test suites
- Cannot test various JVMs at the same time

Tools - Apache JMeter

- The **Apache JMeter™** application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.
- It was originally designed for testing Web Applications but has since expanded to other test functions.
- Run on any environment /workstation that accepts a JVM
- JMeter simulates a group of users sending requests to a target server, and return statistics information of target server through graphical diagrams

Elements or the different components of JMeter are

- Thread Group
- Samplers
- Listeners
- Configuration

Tools - Elements in Apache JMeter

The different components of JMeter are called Elements:

- Thread Group
- Samplers
- Listeners
- Configuration

SOFTWARE TESTING

Tools - Advantages Apache JMeter

- Open source
- Friendly UI
- Platform Independent
- Multi-threading framework
- Highly extensible
- Visualize test result
- Unlimited testing capabilities, Easy installation, Support multi protocol

SOFTWARE TESTING

Tools – Shortcomings of JMeter

- Scripting in JMeter requires some level of expertise and understanding of JMeter elements, regular expressions, session handling etc.
- It does not have a network visualization feature unlike other performance testing tools like Loadrunner.
- A single normal configuration machine is not sufficient for carrying out load test with a large number of users. In such cases either a very high configuration cloud machine is required or distributed testing is performed.
- It does not support ajax, javascript and flash, neither it renders web elements like a browser.
- It provides very limited real-time test monitoring capability as compared to other tools.

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

SOFTWARE MAINTENANCE

Unit 4

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE MAINTENANCE

Software Maintenance Introduction

Dr. Phalachandra H. L
Department of Computer Science and Engineering

SOFTWARE MAINTENANCE

Introduction :

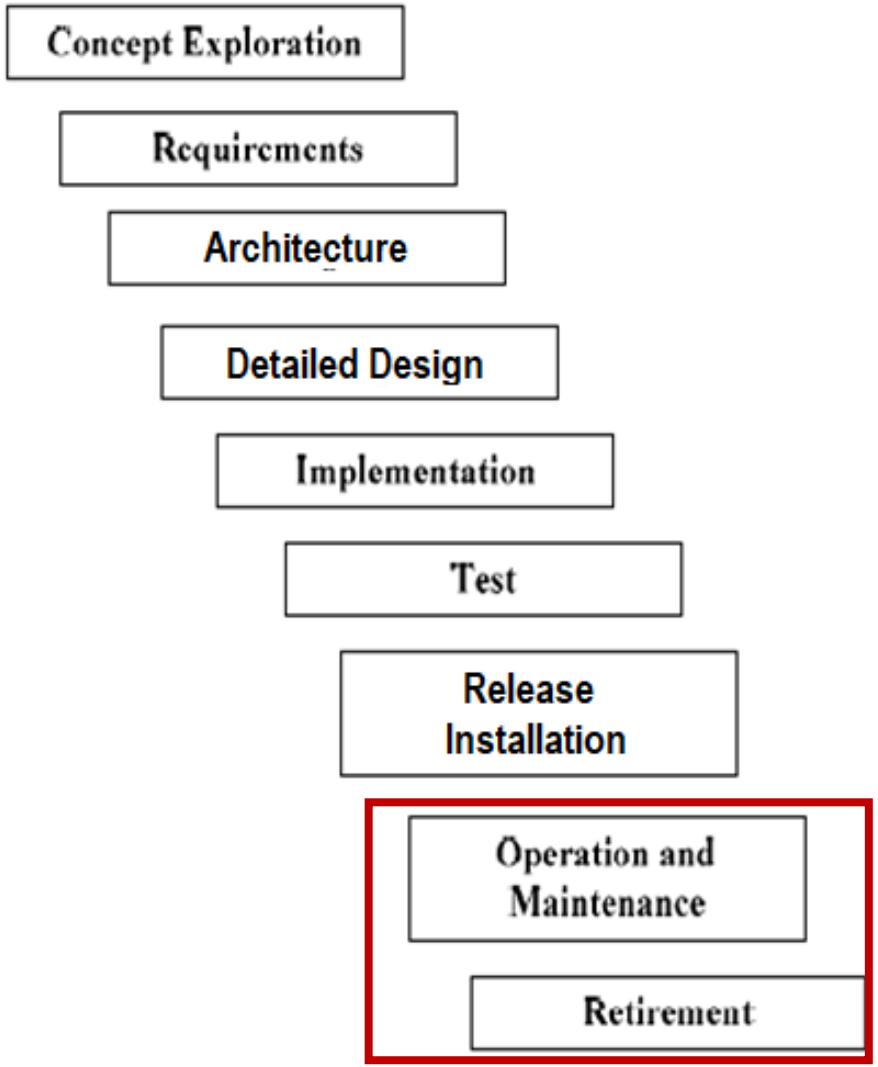
- Software today are large, complex and they are built to last for a long time.
- Typically, the software delivered to the public will go through a large number of modification & enhancements as a result of:
  - Fixing defects reported from the field
  - Correcting requirement and design errors
  - Implementing enhancements
  - Improving performance
  - Adapting the system in response to the environmental changes such as:
    - Hardware changes
    - Software changes within programs that it interface with
    - Network/telecommunication changes
    - Changes in standards (government, industry)
  - Retiring outdated systems
  - Interfacing with other software

SOFTWARE MAINTENANCE

Introduction :

- As discussed in Unit 1, a software system as part of a SDLC progresses through a series of phases as seen
- Maintenance is typically the largest portion of the total life cycle as products are used for a long period of time and this also includes discontinuance and retirement.

Waterfall Model of Software Life Cycle



SOFTWARE MAINTENANCE

Introduction : Formal Definition of Terminology

'Maintenance' – is the sustaining process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [IEEE Std. 610.12-1990]

Other definitions of maintenance:

"the totality of activities required to provide cost-effective support to software" [SWEBOK]

Software maintenance is defined as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment [IEEE 12119]

SOFTWARE MAINTENANCE

Why and who maintains Software

Why

- To ensure that the software product continues to satisfy its users needs.
- Commitment to customers to support software systems which they have invested into

Who

- **Maintainer** could be an organization or a person responsible for carrying out maintenance activities. E.g.
  - Lot of product organizations outsource maintenance to other service organizations like TCS, Wipro, Infosys etc.
  - Engineers who are assigned to maintain.

SOFTWARE MAINTENANCE

Maintenance Activities expected and Actions which can help towards it

Activities Expected

- Maintaining control over the software's day to day functions
- Maintaining control over software modifications
- Perfecting existing functions
- Preventing software performance from degrading to unacceptable levels

Actions which will support it

- Understanding the product, module which needs to be maintained by studying the architecture, design, code, test cases and documentation
- Discussion with architects and developers
- Enhance documentation, Create branches, instrumentation & debug code
- Look at present and the change request at hand, to identify how to satisfy the task at hand
- Look at future and analyze the consequences of its solution as to how it affects any other future maintenance activities

SOFTWARE MAINTENANCE

Maintenance Costs

- Maintenance consumes a major share of software lifecycle financial resources
- Typically ~70% of the total effort spent on the product would be on Maintenance and the rest 30% on Development. The total costs involved would also follow this.
- Some of the technical and non-technical factors affecting software maintenance costs are:
  - Application type.
  - Software novelty.
  - Software maintenance staff availability.
  - Software lifespan.
  - Hardware characteristics.
  - Quality of the software design, construction, documentation, and testing

SOFTWARE MAINTENANCE

Categories of Maintenance

Maintenance is

Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults

Modification of a software product after delivery to improve performance or maintainability

Modification of Software	Correction	Enhancement
When done Proactively	Preventive Maintenance	Perfective Maintenance
When done Reactively	Corrective Maintenance	Adaptive Maintenance

Can be for

or for

Reactive modification of a software product performed after delivery to correct discovered problems

Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment

SOFTWARE MAINTENANCE

Key Issues in Software Maintenance :

Technical

- Code and Documentation Quality
- Limited Understanding –Most times newer and lower skilled people
- Impact Analysis on whether the change is scoped & understood properly

Management

- Staffing and retaining people with right skill levels
- Alignment with economic objectives as maintenance does not have as much direct RoI
- Outsourcing the maintenance to another organization within the organizations or outside the organization have challenges like :
  - Protection of the intellectual property
  - Control over the development process and quality control
  - Learning curve for the product under maintenance
  - Scope of maintenance

Cost

- Application type
- Staff availability
- Lifespan of the system
- Quality of Software artifacts

Predictability

- Measures related to Schedule, time required for maintenance, Complexity and Quality of product

1. Implementation of Processes

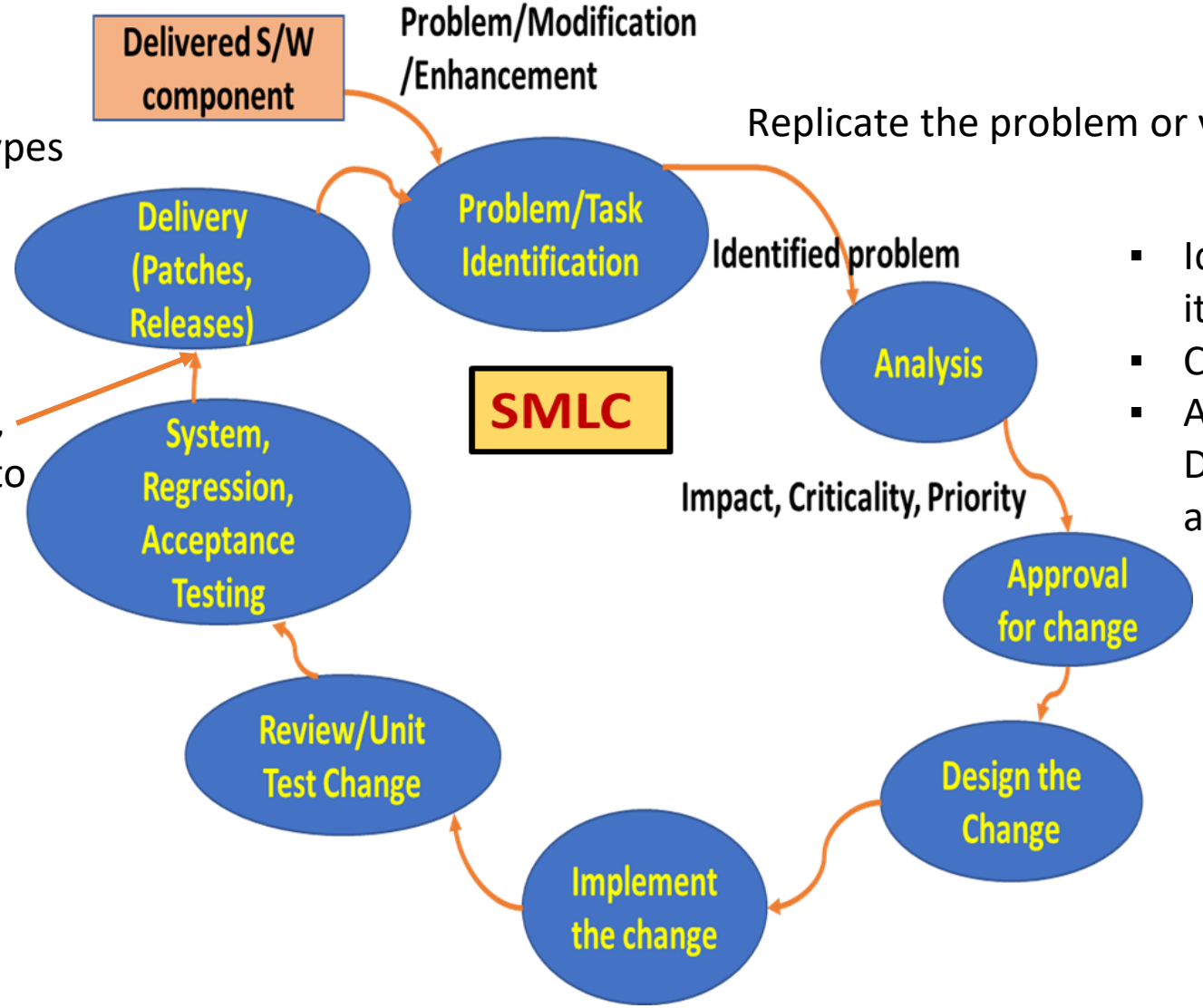
- Develop, document, and execute maintenance process plans including usage of tools as part of the maintenance process (including procedures for handling user reports and modification requests, tracking of problems, responding to users etc.)

SOFTWARE MAINTENANCE

Software Maintenance Activities : 2. Software Maintenance Life Cycle (SMLC)

Different delivery mechanisms with Maintenance lifecycle
Patches – Different Types
Complete products
Application release bundles

Check in the approved, changed components to the CM system



- Identify all affected configuration items
- Consider modification options
- Analyse the impact of problem
  Document the request, the analysis, and the suggested options

SOFTWARE MAINTENANCE

Maintenance Process Activities (Cont.)

3. Migration

- Develop a migration plan for the fixed or new component and execute that plan. This could be simply
  - Notifying and replacing the product and restarting
  - Applying the patch with restart or no-restart based on the product & kind of patch
  - For a complex product with impactful changes, it might need a more elaborate set of activities in addition like white papers, tools etc.
  - Migration, Verification of Migration, Support for older data based on the agreement

4. Software retirement

- Develop a retirement plan for the system including the timeline, communicate and execute that plan
- Retire the older version and running of the new in parallel and make available retd data as needed
- Dispose of superseded hardware and software including licences, stoppage of contracts etc

Tips for Maintenance

- Reproduce the problem.... Before you start fixing it
- Understand the stack traces
- Check your environment variables
- Write a test case that reproduces the problem .. Even better ..Convert the problem into an automated test. ...
- Know your error codes
- Don't assume things work the way they're meant to. ...
- Be clear in your mind about correct behavior. ...
- Pair program whenever possible .. Two pairs of eyes helps
- Guess and check
- Get your code to help you. ...
  - Print
  - Use logs
  - Comment out code and check
- Learn your debugger's capabilities
- Take breaks to get your mind off and restart
- Fix one problem at a time. ...
- Test, Test and Test
- Consider the bigger picture

Techniques or Approaches for Maintenance

Reverse engineering

Reverse engineering is a passive technique that is used to understand a piece of software prior to re-engineering.

Reverse engineering is the process of recovering specification and design information about the software system from its source code

Over time, an application may have been corrected, adapted and enhanced. As a result, the application can become complex, unstable with changes having unexpected and serious side effects and reverse engineering addresses it.

- Reverse engineering identifies the components of a software product and the interrelationships between the components.
- The purpose of reverse engineering is to create a representation of the software in a different format than the existing code or documentation.
- It does not modify the existing product, nor does it result in a new product.
- Some of the typical products created during the reverse engineering process are called graphs and control flow graphs.

Techniques or Approaches for Maintenance

Re-Engineering

- Re-engineering is the process of modifying the software to make it easier to understand, change and extend. This could be for improving maintainability of a software system at reasonable cost.
- Re-engineering is considered from a maintenance perspective, when the software product is no longer viable to employ standard maintenance techniques to support with reasonable efficiency and productivity.
- Re-engineering may involve refactoring
- The re-engineered code is expected to result in reduction of CPU utilization, readability, usability and performance
- This could involve steps like identifying the current process, document the planned reconstruction, reverse engineer, refactor and reconstruct code and data

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4: SOFTWARE QUALITY

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE QUALITY

Software Quality – Perspectives, Attributes & SQA

Dr. Phalachandra H. L
Department of Computer Science and Engineering

PRODUCT QUALITY ATTRIBUTES

Software Quality - Context

- Software systems are complex and evolve or change leading to a need for continuous assessment and evaluation of quality for a satisfactory product
- Increased proliferation of software into our everyday life has led to a scenario where bad quality of software could impact our experiences in life to an extent of loss of life.
- Software quality issues can lead to dissatisfied customers which can impact the businesses.
- Software Quality enhances the long term profitability of the products

PRODUCT QUALITY ATTRIBUTES

Perspectives of Quality

- “Exceeded normal expectations” **(Transcendent)**
- “fitness for use” **(User-based)**
- “based on attributes of the software” **(Product-based)**
- “conformance to specs” **(Manufacturing-based)**
- “Value-based” **(balancing time and cost vs profits)**

PRODUCT QUALITY ATTRIBUTES

Approaches to look at Quality

- Quality of a software product can be visualized as Quality of the product v/s Quality of the process used for building the product
- Quality actions could include checking whether (product or process) conforms to certain norms – Verification & Validation and Audits
- Improvement in quality (if necessary) could be achieved by improving the product &/or process

Relevant Standards

- There have been number of standards established from ISO towards management of quality
- SQA processes provide a means of reviewing and auditing the software

	Conformance	Improvements
Product	ISO 9126	Best Practices
Process	ISO 9001	CMM CMMi SPICE Bootstrap

PRODUCT QUALITY ATTRIBUTES

Software product quality perspectives

Product operation Perspective

Correctness	does it do what I want?
Reliability	does it do it accurately all of the time intended?
Efficiency	will it run on my hardware as well as it can?
Integrity	is it secure?
Usability	can I use it?
Functionality	Does it have all the features which I need?
Availability	Will the product run or be available all the time its needed?

Product revision Perspective

Maintainability	can I fix it?
Testability	can I test it?
Flexibility	can I change it?

PRODUCT QUALITY ATTRIBUTES

Software product quality perspectives (Contd.)

Product transition Perspective

Portability	will I be able to use it on another machine?
Reusability	will I be able to reuse some of the software?
Interoperability	will I be able to interface it with another system?

Overall Environment perspective

Responsiveness	will I be change quickly respond to changes?
Predictability	will I be able to predict the progress?
Productivity	will things be done in the most efficient fashion?
People	will my customers be satisfied
	will my employees be gainfully engaged

PRODUCT QUALITY ATTRIBUTES

Example measures of few attributes

Correctness – degree to which a program operates according to specification

- Defects/KLOC
- Failures/hours of operation

Maintainability – degree to which a program is open to change

- Mean time to change
- Change request to new version (Analyze, design etc.)
- Cost to correct

Integrity - degree to which a program is resistant to outside attack

- Fault tolerance, security & threats

Usability – easiness to use

- Training time, skill level necessary to use, Increase in productivity, subjective questionnaire or controlled experiment

PRODUCT QUALITY ATTRIBUTES

Quality attributes & terminologies of a Product

Quality of products can also be based on the functional and non-functional requirements looked at using the FLURPS+ Model

Functionality - features of system

Localization - Localizable to the local language

Usability - Aesthesis, Intutivity, documentation, typically human factors

Reliability - Frequency of failure in the intended time and availability also gets included into this group

Performance - Speed, throughput, resource consumption

Supportability – serviceability, maintainability

… **E (Extensibility ..)** etc.

SOFTWARE QUALITY PLAN

Software Quality Assurance (SQA)

Software Quality Assurance (SQA) includes the means of monitoring the software engineering processes, and methods used to ensure quality

SQA will involve planning including setting up of goals, commitments, activities, measurements, and verifications

SQA encompasses

- The entire software development processes and activities
- Planning oversight, record keeping, analysis and reporting
- Auditing designated software work products to verify compliance the defined process
- Ensuring that deviations in software work and products according to a documented procedure are recorded and any noncompliance are reported to senior management

SOFTWARE QUALITY PLAN

Who are involved in Software quality assurance (SQA) activities?

- Project managers
  - Establish processes and methods for the product and plan and provide oversight for execution
- All stake holders
  - Perform actions as relevant for the quality of the product
- Software engineers
  - apply technical methods and measures
  - conduct formal technical review
  - perform well-planned software testing
- SQA group
  - Quality assurance planning oversight, record keeping, analysis and reporting
  - Serves as the customer's in-house representative

SOFTWARE QUALITY PLAN

Contents of the plan

Quality cannot be plugged in and needs to be happening throughout the lifecycle.

This is looked at holistically using a Quality plan which addresses the following

1. Responsibility Management
2. Document management and control
3. Requirements Scope
4. Design Control
5. Development control and Rigor
6. Testing and Quality Assurance
7. Risks and Mitigation
8. Quality Audits
9. Defect Management
10. Training Requirements

SOFTWARE QUALITY PLAN

The SQ Plan [IEEE94]

The SQA plan provides a road map for instituting software quality assurance.

Basic items:

- Purpose of plan and its scope
- Management
  - Organization structure, SQA tasks, their placement in the process
  - Roles and responsibilities related to product quality
- Documentation
  - Project documents, models, technical documents, user documents.
- Standards, practices, and conventions
- Reviews and audits
- Test plan and procedure
- Problem reporting, and correction actions
- Tools
- Code control
- Media control
- Supplier control
- Records collection, maintenance, and retention
- training
- risk management

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu

SOFTWARE ENGINEERING

UNIT 4:
METRICS & CMM

Dr. Phalachandra H. L
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet and supplemented by my experience. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

SOFTWARE METRICS

Metrics and Measures in the context of software quality

- Product quality characteristics influences the profitability and customer experiences of businesses
- As considered in the last session, there can be various different perspectives which may need to be considered for the product and processes
- Quantifying this quality is necessary for ensuring that Quality can be measured, contrasted and enhanced as necessary
- Measures and Metrics help in quantifying the quality of the products

Software Metrics : Why Measure

- **Feedback**
  - Quantifies some of the quality attributes and helps in improving software quality
- **Diagnostics**
  - Helps in identifying issues towards quality
  - Evaluates and Establishes productivity impacts of new tools, techniques and trends over time
- **Forecasting**
  - Supports estimating, budgeting, costing and scheduling of future projects
  - Forecast future staffing needs
  - Anticipate and reduce future maintenance needs

SOFTWARE METRICS

Software Metrics : Definitions

- ***Attribute***
  - Is a measurable physical or abstract property of an entity
- ***Measure***
  - Quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process. (a unit used for some attribute)

    E.g. Number of errors
- ***Metric***
  - Quantitative measure of degree to which a system, component or process possesses a given attribute.
  - It's the calculation between two measures

  E.g. Number of errors found per person hours expended

Software Metrics : Traditional Context of Metrics

Progressive Context of Metrics

Not everything that counts can be counted

- Albert Einstein

Agile .. and Values (cultural change)

Trust is critical in this culture

Summarily - Metrics should supplement and not replace thinking

SOFTWARE METRICS

Characteristics of Software Measures

Quantitative:

Metrics should be quantitative and expressible in values.

Understandable:

Metric computation should be easily understood and be clearly defined.

Applicability:

Should be applicable in all phases of development of the software.

Repeatable:

Metric values should be same when measured repeatedly and will need to be consistent.

Economical:

Computation of metric should be economical.

Language Independent:

Metrics should not depend on any programming language

Software Metrics : Categorization of Measures

These metrics could be

- **Direct Measures (internal attributes)**
  - This depends only on the value of the attribute.
  - This is considered as valid and other attributes are measured with respect to this
  - Cost, effort, LOC, duration of testing, number of defects, speed
- **Indirect Measures (external attributes)**
  - These could be derived from the direct measures
  - Defect density, programmers productivity

SOFTWARE METRICS

Software Metrics : Categorization of Measures

Size Oriented

- Size of the software produced

  LOC - Lines Of Code
  KLOC - 1000 Lines Of Code or *SLOC* – Statement Lines of Code (ignore whitespace)

- Typical Metrics:

  Errors/KLOC, Defects/KLOC, Cost/LOC, Documentation Pages/KLOC

Complexity Oriented

- LOC - a function of complexity
- Fan-In, Fan-Out
- Halstead's Software Science (entropy measures)
  - Considering implementation of an algorithm to be collection of token, this approach counts the tokens determining which are operators and which are operands.
  - Number of metrics can be considered like the Hallstead's program length, volume, vocabulary etc.

SOFTWARE METRICS

Software Metrics : Can be product metrics - process metrics – Project metrics

- **Product Metrics could be for**
  - Assessing the state of the project
  - Tracking potential risks
  - Uncovering problem areas
  - Adjusting workflow or tasks
  - Evaluating teams ability to control quality
- **Processes Metrics could be for**
  - Gaining Insights of software engineering tasks (design etc.), work product, or milestones
  - Usage for long term process improvements
- **Project Metrics could be for**
  - Number of software developer
  - Staffing pattern over the life cycle of software
  - Cost and schedule
  - Productivity

Software Metrics : How do we use Metrics

1. Understand the environment/product
2. Formulate &/or Select appropriate metrics for problem
3. Educate
4. Collect
5. Analyze
6. Interpret

} For feedback, diagnostics, forecasting as interested

7. Course-correction
8. Feedback

CMM (CAPABILITY MATURITY MODEL)

Background

- Developed by Software Engineering Institute of Carnegie Mellon University
- Originally intended as a tool for objectively assessing the Capability (ability) of a vendor to deliver software
- A maturity model can be viewed as a set of structured levels that describe how well the behaviors, practices and processes of an organization can reliably and sustainably produce required outcomes.
- This describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature disciplined one
- These models are used as a benchmark for comparison of the software development processes and as an aid to understanding.

CMM (CAPABILITY MATURITY MODEL)

Characterizing CMM Process terminologies

Process

Set of activities, methods, practices and transformations that people use to develop and maintain software

Process Capability

It's the ability of the process to meet specifications. It indicates the range of expected results that can be achieved by following a process. A predictor of future project outcomes

Process Performance

A measure of the actual results for the specific activity achieved from following a processes

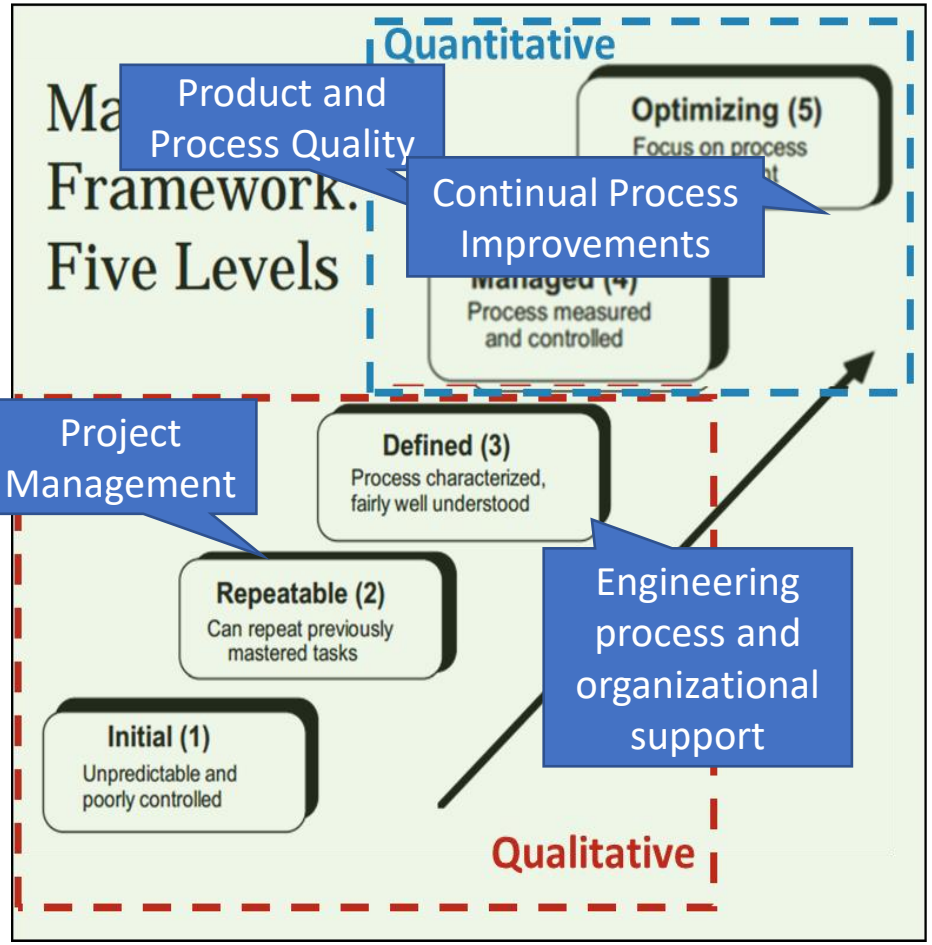
Process Maturity

The extent to which a specific process is explicitly defined, managed, measured, controlled and is effective

CMM (CAPABILITY MATURITY MODEL)

Aspects of the CMM Model

Maturity Levels: a 5-level process maturity continuum - where the uppermost (5th) level is a notional ideal state where processes would be systematically managed by a combination of process optimization and continuous process improvement.



Process maturity perspective as

- Initial (Just do it)
- Repeatable (focus on project management)
- Well Defined (organized assets)
- Analyzed, Improved and Managed (quantitative control)
- Improved and Optimized (Continuously Improving)

Organization maturity perspective as

- Work accomplished according to plan
- Practices consistent with processes
- Processes updated as necessary
- Well-defined roles/responsibilities
- Inter-group communication and coordination
- Management formally commits

CMM (CAPABILITY MATURITY MODEL)

Benefits of Model Based Approach

- Establishes a common language and vision
- Build on a set of processes and practices developed with input from a broad section of the software community
- Provide a framework for prioritizing actions
- Provide a framework for performing reliable and consistent appraisals
- Support industry wide comparisons

CMM (CAPABILITY MATURITY MODEL)

Risks & Limitations of Model based approach

Risks of Model based approach

- Models are simplification of the real world
- Models are not comprehensive
- Interpretation and tailoring must be aligned to business objectives
- Judgement is necessary to use models correctly with insight

Limitations of CMM

- CMM does not specify a particular way of achieving those goals
- CMM can only helps if it is put into place early in the software development process
- CMM is concerned with the improvement of management related activities, which may not be the only area to focus

THANK YOU

Dr. Phalachandra H.L.
Department of Computer Science and Engineering
phalachandra@pes.edu