UNIT-IV
KCS-601
SOFTWARE ENGINEERING
PART-II

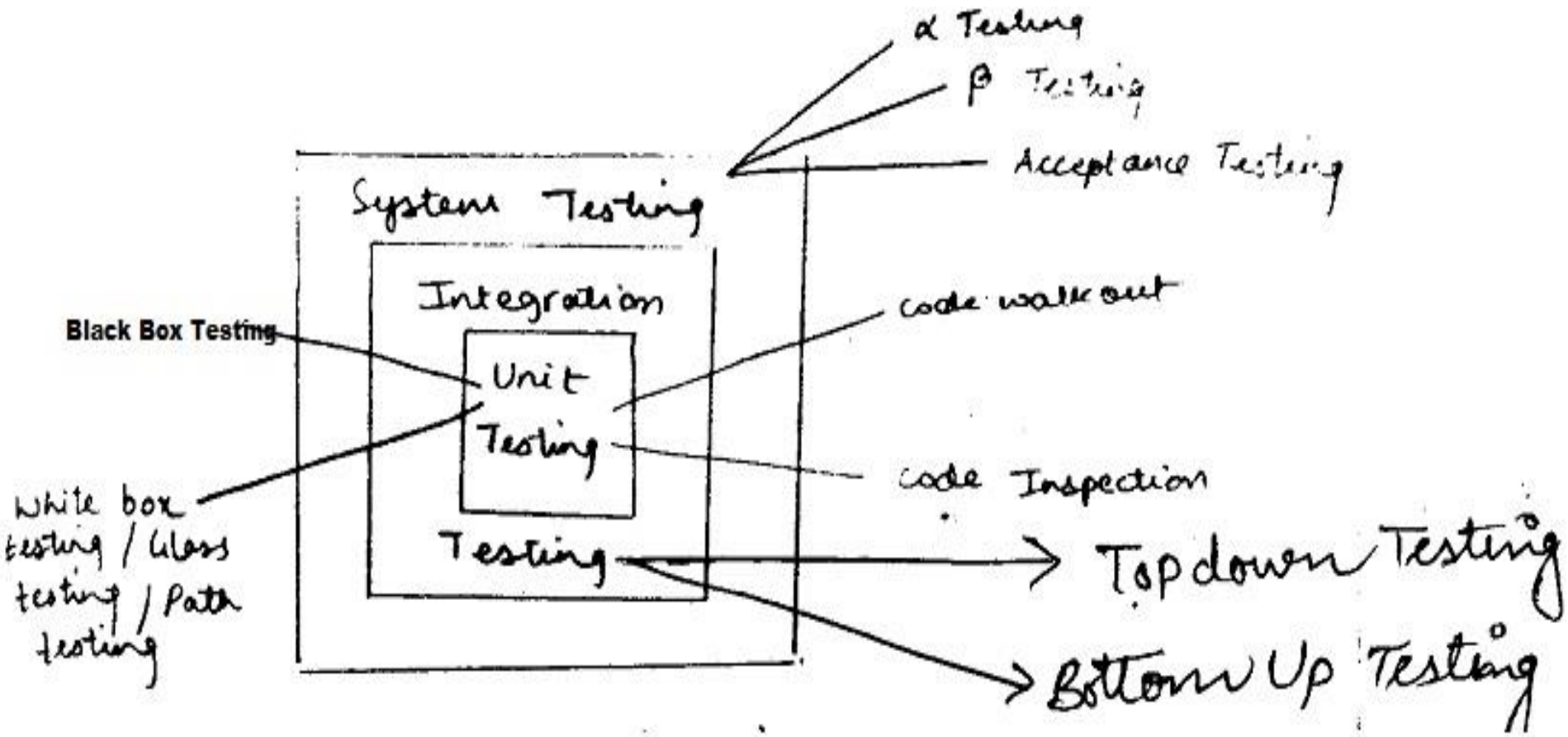
Syllabus

Unit-IV: Software Testing

Testing Objectives, Unit Testing, Integration Testing, Acceptance Testing, Regression Testing, Testing for Functionality and Testing for Performance, Top-Down and Bottom-Up Testing Strategies: Test Drivers and Test Stubs, Structural Testing (White Box Testing), Functional Testing (Black Box Testing), Test Data Suit Preparation, Alpha and Beta Testing of Products.

Static Testing Strategies: Formal Technical Reviews (Peer Reviews), Walk Through, Code Inspection, Compliance with Design and Coding Standards.

Testing Types



Static Testing Strategies

- Static testing is the systematic examination of a program structure for the purpose of showing that certain properties are true regardless of the execution path the program may take.
- Consequently, some static analyses can be used to demonstrate the absence of some faults from a program.
- Static testing represents actual behavior with a model based upon the program's semantic features and structure.
- Human comparison often consists of people exercising little discipline in comparing their code against notions of intent that are only loosely and imprecisely defined.

Static Testing Strategies

- **Static testing strategies include:**
  - Formal Technical Reviews(Peer Reviews).
  - Desk-checking.
  - Code-Walkthroughs.
  - Code inspections.

Software Reviews

- **A Review can be defined as:**
  - "A meeting at which the software element is presented to project personnel, managers, users, customers, or other interested parties for comment or approval."
- **What is a software review?**
  - A software review can be defined as a filter for the software-engineering process.
  - The purpose of any review is to **discover errors in the analysis, design, and coding, testing and implementation phases** of the software-development cycle.
  - The other purpose of a review is to see whether procedures are applied uniformly and in a manageable manner.

Software Reviews

- **Objectives for Reviews:**
  - To ensure that the software elements conform to their specifications.
  - To ensure that the development of the software element is being done as per plans, standards, and guidelines applicable for the project.
  - To ensure that the changes to the software elements are properly implemented and affect only those system areas identified by the change specification.
- **Types of Reviews:**
  - **Informal Technical Review:** An informal meeting and informal desk checking.
  - **Formal Technical Review:** A formal software quality assurance activity through various approaches, such as structured walkthroughs, inspections, etc.

What is a Formal Technical Review?

- A formal technical review is a **software quality assurance activity** performed by software-engineering practitioners to improve software product quality.
- The product is scrutinized for completeness, correctness, consistency, technical feasibility, efficiency, and adherence to established standards and guidelines by the client organization.
- The FTR serves as a training ground, enabling junior engineers to **observe different approaches to software analysis, design, and implementation.**
- Each FTR is conducted as a meeting and will be successful only if it is properly planned, controlled, and attended.

Objectives of a Formal Technical Review

- **The various objectives of a formal technical review are as follows:**
  - To uncover errors in logic or implementation.
  - To ensure that the software has been represented according to predefined standards.
  - To ensure that the software under review meets the requirements.
  - To make the project more manageable.
- **For the success of a formal technical review, the following are expected:**
  - The schedule of the meeting and its agenda reach the members well in advance.
  - Members review the material and its distribution.
  - The reviewer must review the material in advance.

The Peer-Review Meeting

- The Meeting should consist of **two to five people** and should be restricted to **not more than two hours** (preferably).
- The aim of the review is **to review the product/work and the performance of people**.
- When the Product is ready, the developer **informs the project leader** about the completion of the product and **requests for review**.
- The **Project Leader Contacts** the **review leader** for the review.
- The **Review Leader** asks the reviewer to **Perform an Independent Review** of the **product/work before the Scheduled FTR.**

Results of FTR

- Meeting decision
  1. Whether to accept the product/work without any modifications.
  2. Accept the product/work with certain changes.
  3. Reject the product/work due to error.
- Review summary report
  1. What was reviewed?
  2. Who reviewed it?
  3. Findings of the review.
  4. Conclusion.

2-Desk Checking, 3-Code-Walkthrough, 4-Code Inspection

Desk checking: It is done manually by the author of the code, desk checking is a method to verify the portions of the code for correctness. Such verification is done by comparing the code with the design or specifications to make sure that the code does what it is supposed to do and effectively.

Code walkthrough: This method and formal inspection (described in the next section) are group-oriented methods. Walkthroughs are less formal than inspections. The line drawn in formalism between walkthroughs and inspections is very thin and varies from organization to organization. The advantage that walkthrough has over desk checking is that it brings multiple perspectives. In walkthroughs, a set of people look at the program code and raise questions for the author. The author explains the logic of the code, and answers the questions. If the author is unable to answer some questions, he or she then takes those questions and finds their answers.

2-Desk Checking, 3-Code-Walkthrough, 4-Code Inspection

Formal inspection: Code inspection-also called Fagan Inspection (named after the original formulator) - is a method, normally with a high degree of formalism. The focus of this method is to detect all faults, violations, and other side-effects. This method increases the number of defects detected by demanding thorough preparation before an inspection/review;

- Enlisting multiple diverse views;
- Assigning specific roles to the multiple participants; and
- Going sequentially through the code in a structured manner.

Roles in Inspection

1. **Author of the code:** Basic goal should be to learn as much as possible with regard to improving the quality of the document
2. **Moderator:** The person who is expected to formally run the inspection according to the process.
3. **Inspectors:** These are the people who actually provides, review comments for the code. There are typically multiple inspectors.
4. **Scribe:** The person who takes detailed notes during the inspection meeting and circulates them to the inspection team after the meeting.

Differences Between Walk-throughs and Inspections/Reviews

- The basic difference between the two is that a **walk-through is less formal** and has only a few steps, whereas **inspections and reviews are more formal** and logically sequential with many steps.
- Both Processes are undertaken before actual development, and hence they are conducted on documents, such as a development plan, RDD and SRS and design document **to examine their authenticity, completeness, correctness, and accuracy.**
- **Both are costly but the cost incurred is comparatively much lower than the cost of repair at a much later stage in the development cycle.**

Programming Errors

- Testing and debugging are important activities in software development.
- Compilation errors
  - Syntax error (example: missing a semi-colon).
  - Semantic error. (For example, applying modulus % on floating-point value for certain programming languages.)
  - Easiest type of errors to fix.
- Runtime errors
  - Occur at runtime.
  - Java's exception mechanism can catch such errors.
- Logic errors
  - Program runs but produces incorrect result.
  - Hard to characterize, hence hardest to fix.
- Programming errors are also known as bugs

Software Bug

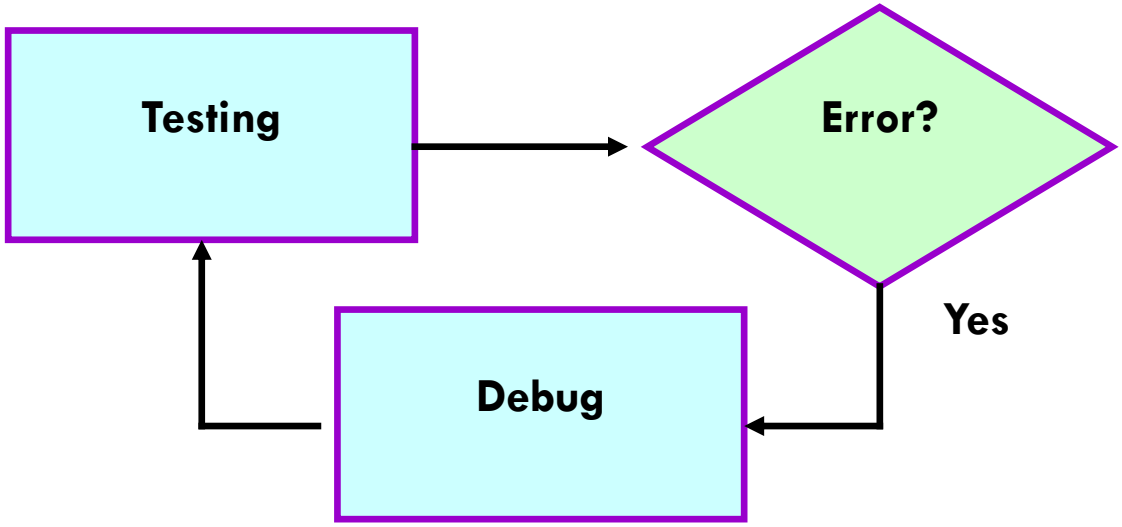
- A **software Bug** is an error, flaw, failure, or fault in a computer program or system that causes it **to produce an incorrect or unexpected result**, or **to behave in unintended ways.**
- Most bugs arise from **mistakes and errors** made by people in either a program's source code or its design, and a few are caused by **compilers producing incorrect code**.
- A program that contains a large number of bugs that seriously interfere with its functionality, is said to be ***buggy***.
- Reports detailing bugs in a program are commonly known as **Bug reports, defect reports, fault reports, problem reports, trouble reports, change requests**, and so forth.

Common Types of Computer Bugs

- **Arithmetic Bugs-**
  - Division by zero.
  - Arithmetic overflow or underflow.
  - Loss of arithmetic precision due to rounding or numerically unstable algorithms.
- **Logic Bugs-**
  - Infinite loops and infinite recursion.
  - Off by one error, counting one too many or too few when looping.
- **Syntax Bugs-**
  - Use of the wrong operator, such as performing assignment instead of equality test.
  - For example, in some languages x=5 will set the value of x to 5 while x==5 will check whether x is currently 5 or some other number.
- **Well-known Bugs-**
  - Year 2000 problem(Y2K bug)

Testing and Debugging

- Testing
  - To determine if a code contains errors.
- Debugging
  - To locate the error and fix it.



DEBUGGING

- **Introduction/Definition-**
  - Debugging Means identifying, locating, and correcting the bugs usually by running the program.
  - It is an extensively used term in programming.
  - These bugs are usually logical errors.

During the compilation phase the source files are accessed and if errors are found, then that file is edited and the corrections are posted in the file. After the errors have been detected and the corrections have been included in the source file, the file is recompiled. This detection of errors and removal of those errors is called debugging. The file is compiled again, so changes done last time get included in the object file also by itself. This process of compilation, debugging, and correction posting in the source file continues until all syntactical errors are removed completely. If a program is very large and complex, the more the program has to be corrected and compiled.

DEBUGGING

- Successful **Compilation of the Program** means that now the program is **following all the rules** of the language and is **ready to execute**.
- All of the **Syntax Errors** of the program are indicated by the complier at this stage.
- **Debugging** is a **logical Process** of **finding and reducing the number of bugs**, **or defects**, in a computer program thus **making it behave as expected.**

Debugging Categories

- **The Various Categories for Debugging are-**
  - Brute-force debugging.
  - Backtracking.
  - Cause elimination.
  - Program slicing.
  - Fault-tree analysis.

Debugging Process

- Debugging is **not testing** but always **Occurs as a Consequence of Testing.**
- Debugging process begins with the E**xecution of a Test Case.**
- Debugging attempts to match Symptom with Cause, thereby **leading to error correction.**
- **Debugging will always have one of two outcomes-**
  - The **Cause will be found** and **Corrected and Removed.**
  - The **Cause will not be found**.

Debugging Process

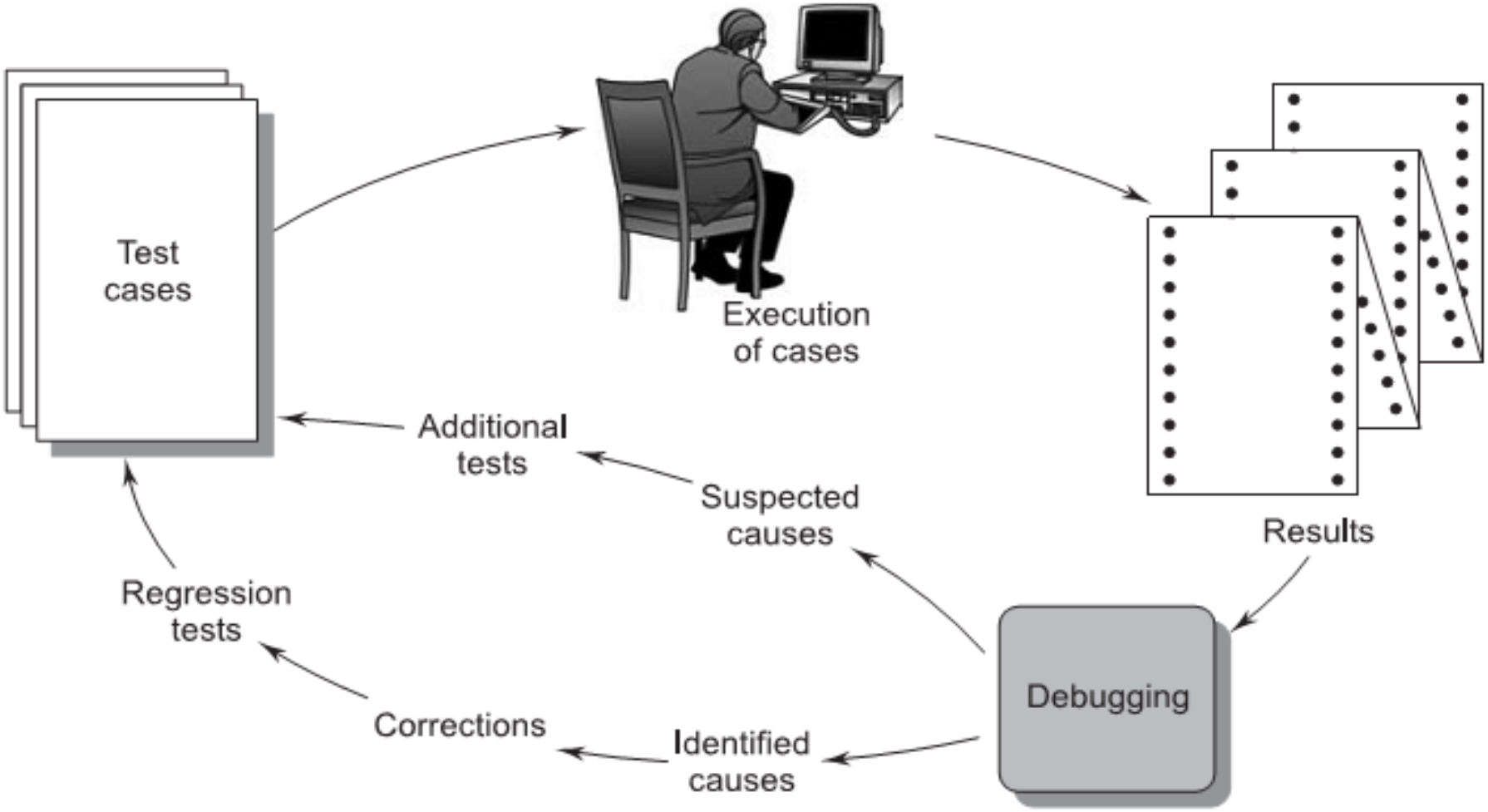


FIGURE The Debugging Process

Program Debugging

- People think that program testing and debugging are the same thing.
- Though closely related, they are **two distinct processes**.
- **Testing establishes** the **presence of errors** in the program.
- Debugging is the **locating of those errors** and **correcting them.**
- Debugging **Depends on the output of testing** which tells the programmer about the **presence or absence of errors.**
- Debugging can be viewed as a **Problem-Solving Process.**
- There is **no standard method** to teach how to debug a program.
- The Debugger must be a **skilled person** who can easily understand the errors by viewing the output.
- The debugger **must have knowledge of common errors**, which occur very often in a program.

Debugging Guidelines

- **Some General guidelines for effective debugging include-**
  - Debugging requires **a thorough understanding of the program design.**
  - Debugging may sometimes even **require a full redesign of the system**.
  - One must be aware of the possibility that any **error correction may introduce new errors.**
  - Therefore, after **every round of error-fixing**, **regression testing must be carried out.**

Characteristics of Bugs

- **Some characteristics of bugs are as follows-**
  - The **Symptom may disappear when another error is corrected**.
  - The **Symptom may be caused by a Human Error.**
  - The **Symptom may be a result of Timing Problems**.
  - It may be **Difficult to Accurately Reproduce input conditions**.
  - The **Symptom may be due to causes that are distributed across a number of tasks running on different processors.**

Testing Vs Debugging

TESTING	DEBUGGING
a) Finding and locating of a defect	a) Fixing that defect
b) Done by Testing Team	b) Done by Development team
c) Intention behind is to find as many defect as possible	c) Intention is to remove those defects

- Purpose: To show that program has bug
- Testing starts with known conditions,uses predefined procedures and has predictable outcomes
- Testing can and should be planned,designed and scheduled

- Purpose: To find the error or misconception that led to the program's failure and to design and implement the program changes that correct the error
- Debugging starts from possibly unknown initial conditions and the end cannot be predicted
- Procedures for and duration of , cannot be planned for debugging