Software Testing and Quality Analysis

Module Supervisor: Mrs. M. Halassa

Chapter 1:
Introduction to test and validation methods

Plan

- Introduction
- what is software ?
- Some known bugs
- Verification and Validation V&V.
- Test and Quality
- Why Testing is Important
- The growth of software testing (history)
- Seven Principle of Testing
- Software Testing Challenges
- Conclusion

Introduction

- The complexity of new developments as well as for IT maintenance makes it impossible to carry out applications without any errors.
- Based on this observation, it is therefore essential not to ignore software **testing** activities.

what is software ?

- Computer **software**, is a collection of computer programs and related data and documents that does a specific job and fills a specific requirement

- **Definition Software (IEEE):**

Computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system.

what is software ?

A software is not just code, but also...

- Needs, expressed by a customer
- Project management

- Life cycle with its derivatives.
- Development tools and environment

- a specification
- Design: global, architectural, detailed
- of the Code
- An executable
- **... Tests !**

Software is a requirement

- Software is almost "everywhere"
- Many things in our daily lives are unimaginable without the software
- Office automation, air travel, Electronic payment, schooling, scientific research, leisure,etc.
- Therefore, our life depends very heavily on the **quality** of the software that manages it.

Software is a requirement

Some known bugs

- **The 1962 bug:** A space rocket diverted from its trajectory because of a mathematical formula that was poorly transcribed into source code.
- **1983:** In the middle of the Cold War, Soviet surveillance software detected fake ballistic missiles sent from the United States.
- **In 1992**, London ambulances were run by faulty software…People lost their lives…

Some known bugs

- **On April 26, 1994** China Airlines Airbus A308 crashed due to a so are bug, killing 264 innocent lives
- **In 1996**, the Arianne 5 rocket exploded after 30 seconds of flight
  - Error converting between digital data!
  - Cost: $100 million

Some known bugs

- **In may of 1996,** a software bug caused the bank a accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars As you see, testing is important because software bugs could be expensive or even dangerous

- **In 2003** Power failure in the United States and Canada

- Impact on 55 million habitants. $6 billion

As you see, testing is important because software bugs could be expensive or even dangerous

Some known bugs

- **In 2004** a failure of the alarm system of a power plant produced a power outage(electricity cut) in the United States and Canada
  - $6B lost!

⇒ Need to "check" some software/systems to ensure software quality.

Error, fault, Failure

- **Error:** a human action that produces an incorrect result,

e.g. a programming error

- **Defect or bug** : a flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition.
- **Failure:** the physical or functional manifestation of a defect, A defect, if encountered during execution, may cause a failure. Deviation of the component or system from its expected delivery, service or result.

Types of defects:

Severity	Description
Critical	This defect indicates complete shut-down of the process, nothing can proceed further.
Major	It is a highly severe defect and collapses the system. However, certain parts of the system remain functional.
Medium	It causes some undesirable behavior, but the system is still functional.
Low	It won't cause any major break-down of the system.

The sources of software bugs

The origin of computer bugs comes from human errors such as:

- A misunderstanding of the need and requirements of the sponsoring organization
- Lack of know-how and expertise,
- Lack of IT skills to understand the development and execution environment,
- The complexity of the platforms.

What is a successful test ?

Software testing is the most popular method of verifying software. as all statistical studies confirm, the test phase consumes most of the human or material resources (40%, or even 60% of the total cost of the project.

- "A successful test is not a test that found no defects, but a test that actually found a defect (or anomaly)" *G.J.Myers*

What is Software Testing?

- **Several definitions:**

"Testing is the process of establishing confidence that a program or system does what It is supposed to."*by Hetzel 1973*

"Testing is the process of executing a program or system with the intent of finding errors*."* *by Myers 1979*

"Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results "

By Hetzel 1983

Test Definition

- **According to *the IEEE*** "Testing is the execution or evaluation of a system or component, by automatic or manual means, to verify that it meets its specifications or identify differences between expected and achieved results"

Test Definition

- “Software testing is an activity that is part of the development process. It is conducted according to **the rules of quality assurance.**

He is interested in both the source code and the behavior of the software. Its objective is to minimize the chances of an anomaly appearing with automatic or manual means that aim to detect both the various possible anomalies and the possible defects that would cause them ". *Hermes Edition;*

Verification and Validation V&V

The test is often associated with the terms **verification** and **validation**.

- **Verification** is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

- **Validation** is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

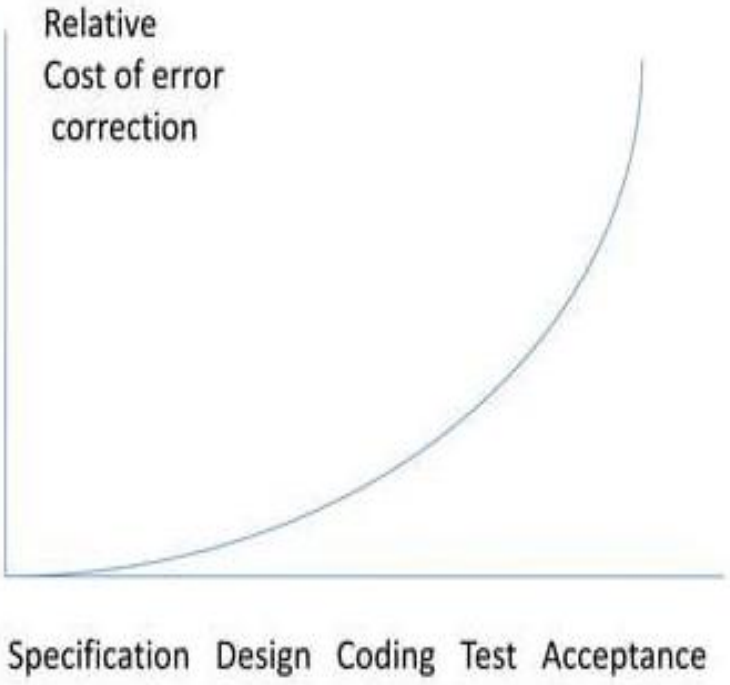
- **Verification: Are we building the product right?** software must comply with its specification
- **Validation : Are we building the right product**? i.e., the software must do what the user needs

Verification and Validation V&V

Cost of an defect depending on when it is detected

-The **costs** of fixing defect is **increase with the time** they remain in the system.

-Detecting errors at an **early stage** allows for error correction at reduced cost.



Why Testing is Important

- Software testing is really required to point out the defects and errors that were made during the development phases.
- It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.
- It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.
- Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.

Why Testing is Important

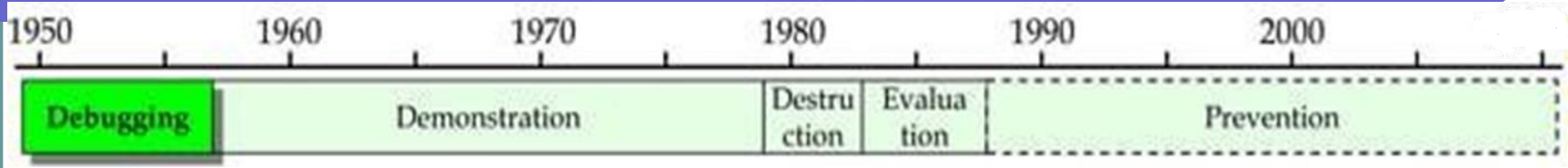
- Testing is required for an effective performance of software application or product.
- It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
- It's required to stay in the business.

Test and Quality

- Using tests, it is possible to measure the quality of software in terms of defects found, for both functional and non-functional characteristics and requirements.
- When tests find defects, the quality of the software system increases when these defects are corrected. By understanding the causes of defects found in other projects, processes can be improved, which can prevent the occurrence of these defects and, as a result, improve the quality of future systems.

This is an aspect of quality assurance.

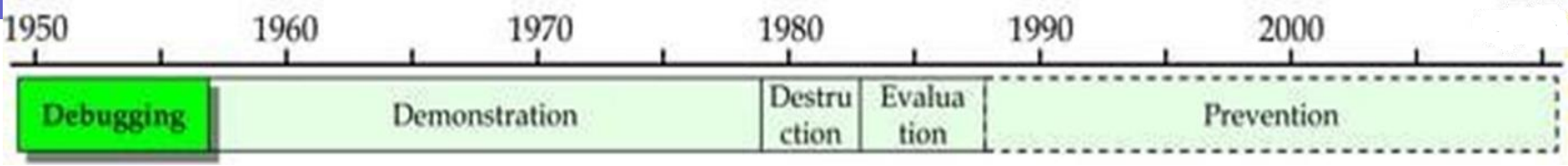
The growth of software testing (history)



1. The Debugging-Oriented Period (-1956)
2. The Demonstration-Oriented Period (1957-1978)
3. The Destruction-Oriented Period (1979-1982)
4. The Evaluation-Oriented Period (1983-1987)
5. The Prevention-Oriented Period (1988-)

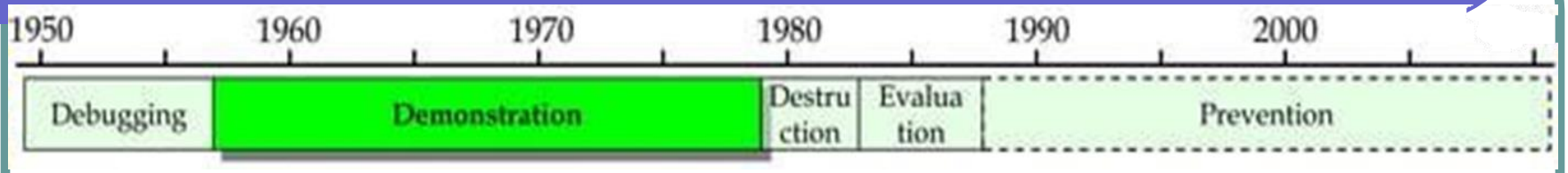
1. Debugging oriented testing

early computing - 1956



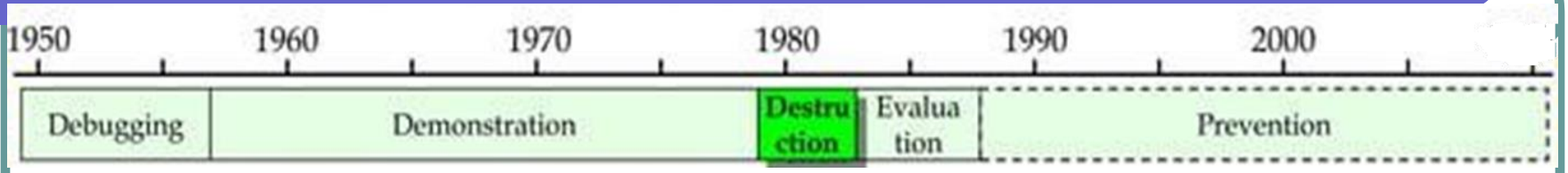
- This is the beginning of computing. The programs were written and checked by the programmers themselves until they were convinced that all errors had been identified and corrected.
- All or part of the error identification and correction activities were commonly referred to as testing.
- The criteria used for the selection of test cases were based on the experience of the programmers and their understanding of the written programs.

2. Demonstration-oriented test
1957-1978



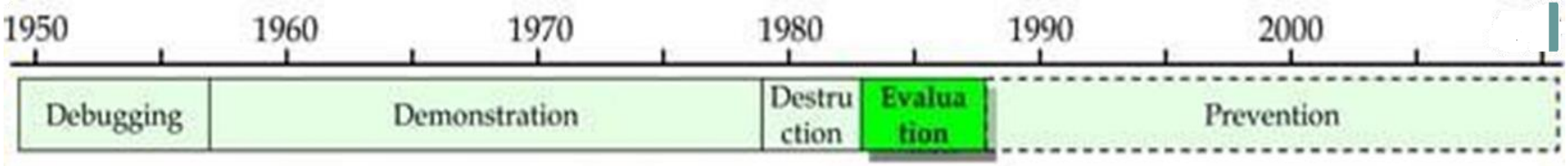
- At that time, the main goal was to prove that the software met the requirements.
- For the first time, testing and debugging activities were identified as separate activities:
  - **Testing**: is about making sure the program does what it's supposed to do.
  - **Debugging:** consists of ensuring that the program runs (does not crash, no total blockage).

3. Destruction oriented test
1979 - 1982



- During this period, the purpose of the test was rather to find errors. This vision was born with the publication of the book *''The Art of Software Testing''* .
- *The distinction is* again drawn between the terms test and debugging.
  - **Test:** relates to the detection of the presence of errors in a program.
  - **Debugging:** concerns the localization and correction of errors.

4. Evaluation-oriented test
1983 - 1987

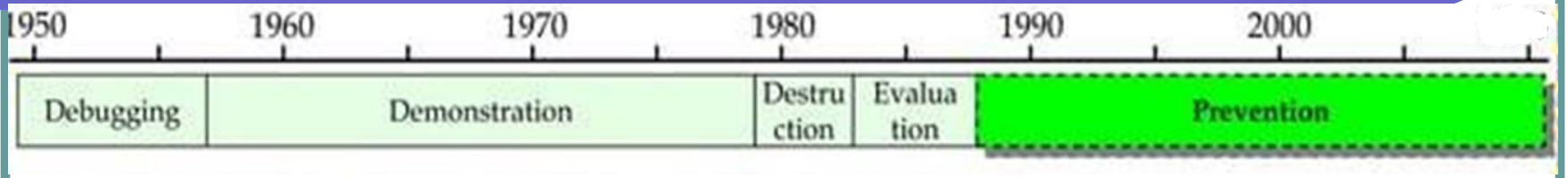


- From now on, the intention was to prove the **quality of the software** and measure it. As with other code analysis and inspection techniques, testing was considered an **evaluation** activity at the end of any software life cycle

- highlighting the principle that rather than an error being detected, the less will be the cost of correcting it.

5. Prevention-oriented test

1988 - so far



- The aim is to prevent design defects before the software product is placed on the market and to ensure that the software product meets the specifications.
- This approach was initiated by Gelperin and Hetzel (*The growth of software testing*), who generalized unit testing methods and developed a comprehensive methodology for practical test management. Their philosophy is to prevent errors at each phase of the life cycle,

A good tester -1-

Curious , perceptive, attentive to detail

- To **comprehend** the practical scenarios of the **customer**
- To be able to analysis the structure of the test
- To **discover details**, where failure might show

has a critical eye

- Test object contain defects- you just have to find them
- Do not believed everything you are told by the developers
- One must not get frightened by the fact that serious defects may often be found which will have impact on the course of the project.

A good tester -2-

Good communication skills

- To bring bad news to the developers
- To overbear frustration state of minds
- Both technical as well as issue of the practical use of the system must be understood and communicated
- Positive communication can help to avoid or to ease difficult situations.
- To quickly establish a working relationship with the developers

Experiences

- **Personal factors** influencing error occurrence
- Experience helps identifying where **errors** might **accumulate**

Difference : to design, to develop , to test

- Testing requires a different mindset from designing developing new computer systems
  - **Common goal**: to provide good software
  - **Design** mission: help the customer to supply the right requirements
  - **Developer's** mission: convert the **requirements** into functions
  - **Tester's** mission: examine the **correct** implementation of the customer's requirements
- In principle, one person can be given all three roles to work at.
  - Differences in goal and role models must be taken into account
  - This is difficult but possible
  - Other solution (independent testers) are often easier and produce better results

Seven Principle of Testing

- **P1:Testing shows presence of defects** Testing can prove the presence of defects, but cannot prove the absence of defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
- **P2: Exhaustive testing is impossible :**Testing all combination s of inputs and preconditions is not feasible. Instead of exhaustive testing, risk analysis, time & cost and priorities should be used to focus testing efforts.
- **P3: Early testing** To find defects early, testing activities shall be started as early as possible in the software or system development life cycle, and shall be focused on defined objectives.

Seven Principle of Testing

- **P4: Defect clustering** Testing effort shall be focused proportionally to the expected and later observed defect density of modules. A small number of modules usually contains most of the defects discovered during prerelease testing, or is responsible for most of the operational failures.

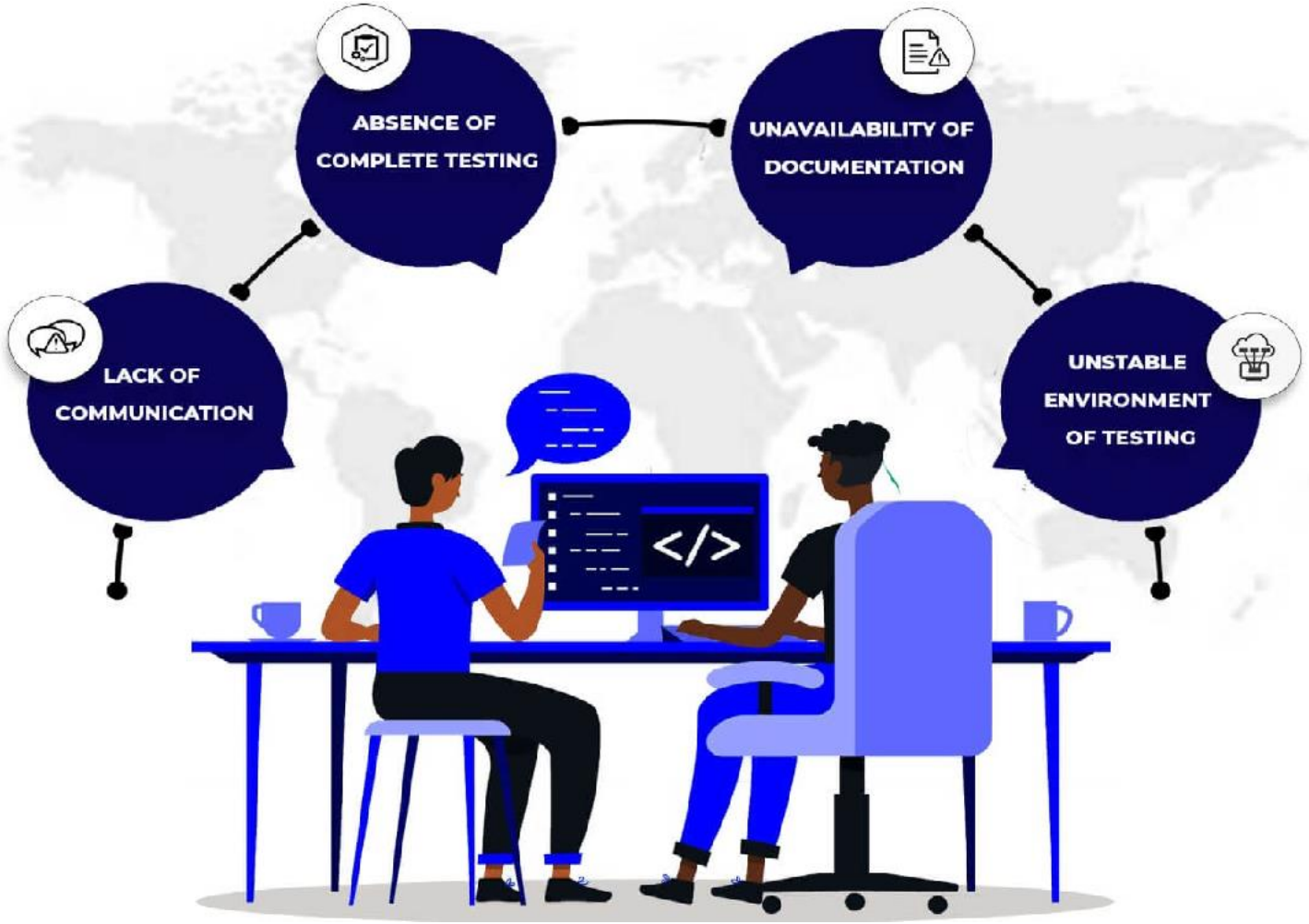
- **P5: Pesticide paradox** If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this "pesticide paradox", test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or stem to find potentially more defects.

Seven Principle of Testing:

- **P6: Testing is context dependent** Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e- commerce site.

- **P7: Absence-of-errors fallacy**
Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

Software Testing Challenges and Their Solutions



Software Testing Challenges and Their Solutions

Lack of communication: Communication gaps can be due to various factors, including shift gaps, different time zones of clients and developers, misinterpretations,and many more. It may be between the customer and project manager or between team members, and it impacts the quality of the product

Solution:

Development and testing teams must collaborate at regular intervals. Discussions at regular intervals maintain a transparent process and also help team members stay aligned on their deliverables

Software Testing Challenges and Their Solutions

- **Absence of Complete testing:** There are a few million lines of code. There are chances of errors. There is high demand for software applications in shorter times, so software companies face the challenges of testing the whole process.
- **Soliution :** A team believes in doing the testing process step by step, and so they should prioritize the list of requirements in identifying testing areas to know which tests are to be carried out earlier.

Software Testing Challenges and Their Solutions

Unavailability of documentation: One of the challenges in software testing is incomplete or missing documentation. This can make it difficult to understand the requirements or to know what needs to be tested.

Solution It is important to have complete and up-to-date documentation for the software under test. This documentation should be easily accessible by the testing team

Software Testing Challenges and Their Solutions

•**Unstable environment of testing:** With the changing requirement specification, developers also make changes in the test environments to fix the issues. When multiple testers are involved in testing a single product, it becomes tedious to track all the changes made by an individual team member.

•. **Solution:** The software might need to be tested in different environments

Conclusion

- Software testing is an activity that is part of the development process. It is conducted according to **the rules of quality assurance**.
- Its objective is to minimize the chances of an anomaly appearing.