TESTING STRATEGIES

LECTURE # 30

LECTURE BY ENGR. SIDRA

VALIDATION TESTING

- Validation testing begins at the culmination of integration testing, when individual components have been exercised, the software is completely assembled as a package, and interfacing errors have been uncovered and corrected.
- Testing focuses on user-visible actions and user-recognizable output from the system.
- Validation succeeds when software functions in a manner that can be reasonably expected by the customer.

VALIDATION TESTING

- After each validation test case has been conducted, one of two possible conditions exists:
  - The function or performance characteristic conforms to specification and is accepted or
  - A deviation from specification is uncovered and a deficiency list is created.
- Deviations or errors discovered at this stage in a project can rarely be corrected prior to scheduled delivery.
- It is often necessary to negotiate with the customer to establish a method for resolving deficiencies.

ALPHA / BETA TESTING

- Alpha Testing:
  - It is conducted at the developer's site by a representative group of end users.
  - The software is used in a natural setting with the developer "looking over the shoulder" of the users and recording errors and usage problems.
  - Alpha tests are conducted in a controlled environment.
- Beta Testing:
  - It is conducted at one or more end-user sites.
  - The beta test is a "live" application of the software in an environment that cannot be controlled by the developer.
  - The customer records all that are encountered during beta testing and reports these to the developer at regular intervals.
  - As a result of problems reported during beta tests, you make modifications and then prepare for release of the software product to the entire customer base.

SYSTEM TESTING

- System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform allocated functions.
- Types of System tests:
  - Recovery testing
  - Security testing
  - Stress testing
  - Performance testing
  - Deployment testing

SYSTEM TESTING

- Recovery testing:
  - It is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
  - If recovery is automatic (performed by the system itself), reinitialization, checkpointing mechanisms, data recovery, and restart are evaluated for correctness.
  - If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits.
- Security testing:
  - Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.

SYSTEM TESTING

- Stress testing:
  - Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.
  - For example: Special tests may be designed that generate ten interrupts per second, when one or two is the average rate
- Performance testing:
  - It is designed to test the run-time performance of software within the context of an integrated system.
  - Used to evaluate system performance under normal and heavy usage
  - E.g., how long should a task take?

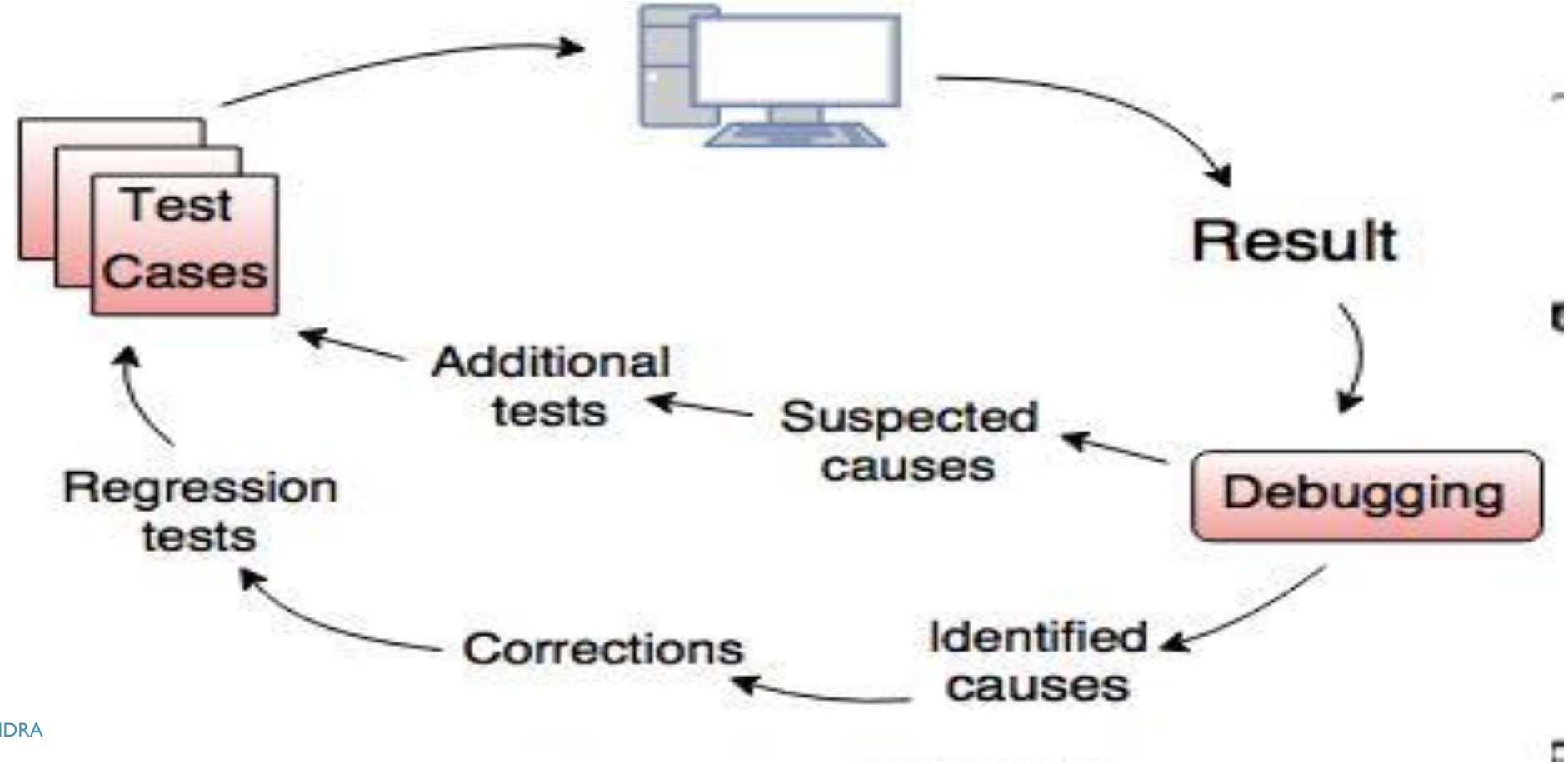
SYSTEM TESTING

- Deployment testing:
  - Sometimes called configuration testing,
  - It exercises the software in each environment in which it is to operate.
  - In addition, deployment testing examines all installation procedures and specialized installation software (e.g., "installers") that will be used by customers, and all documentation that will be used to introduce the software to end users.

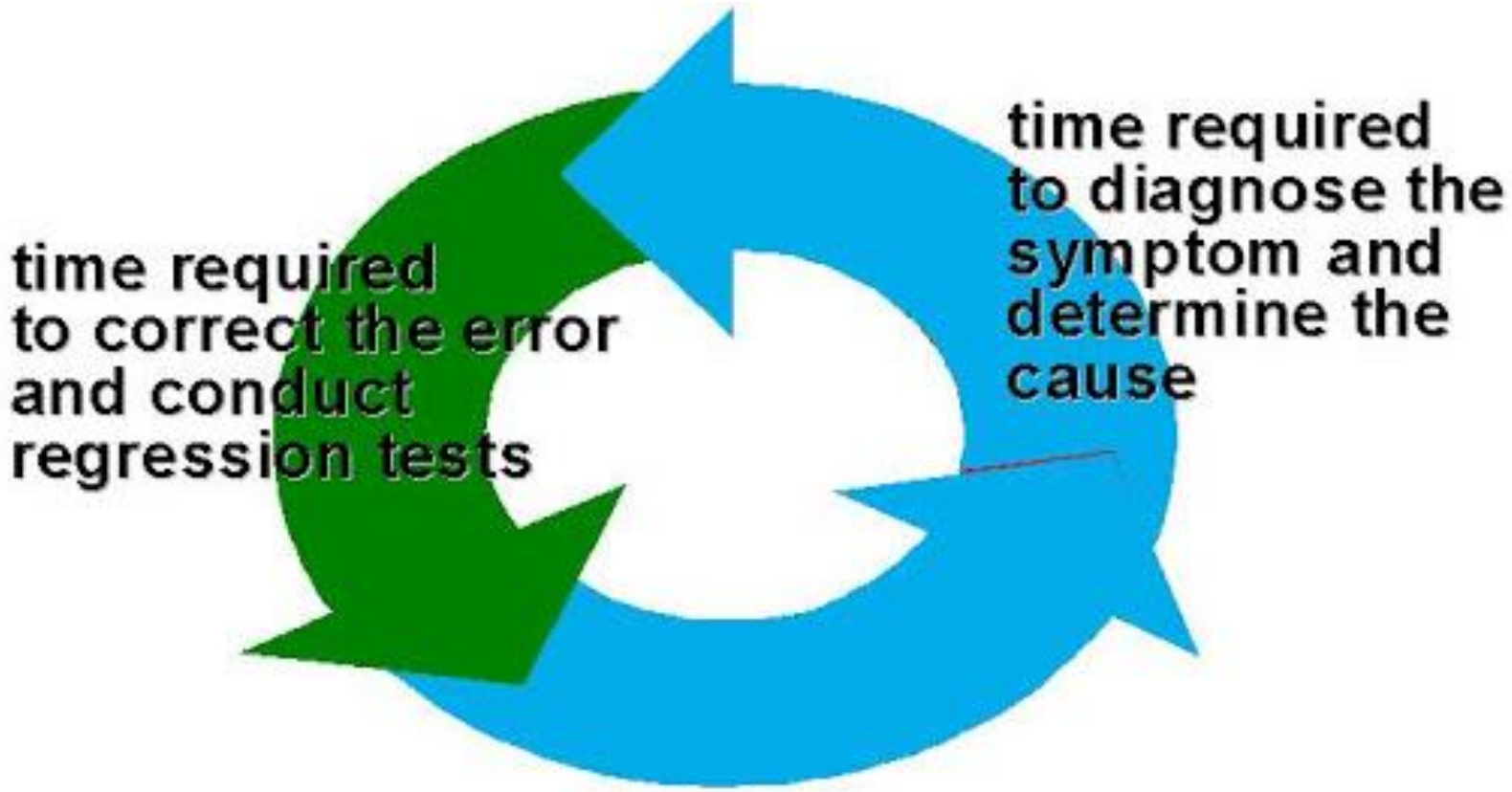
DEBUGGING

- Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is the process that results in the removal of the error.

DEBUGGING PROCESS



DEBUGGING EFFORT



CORRECTING THE ERROR

- Is the cause of the bug reproduced in another part of the program?
  - In many situations, a program defect is caused by an erroneous pattern of logic that may be reproduced elsewhere.
- What "next bug" might be introduced by the fix I'm about to make?
  - Before the correction is made, the source code (or, better, the design) should be evaluated to assess coupling of logic and data structures.
- What could we have done to prevent this bug in the first place?
  - If you correct the process as well as the product, the bug will be removed from the current program and may be eliminated from all future programs.