Software Quality Assurance and Testing
Lecture - 06

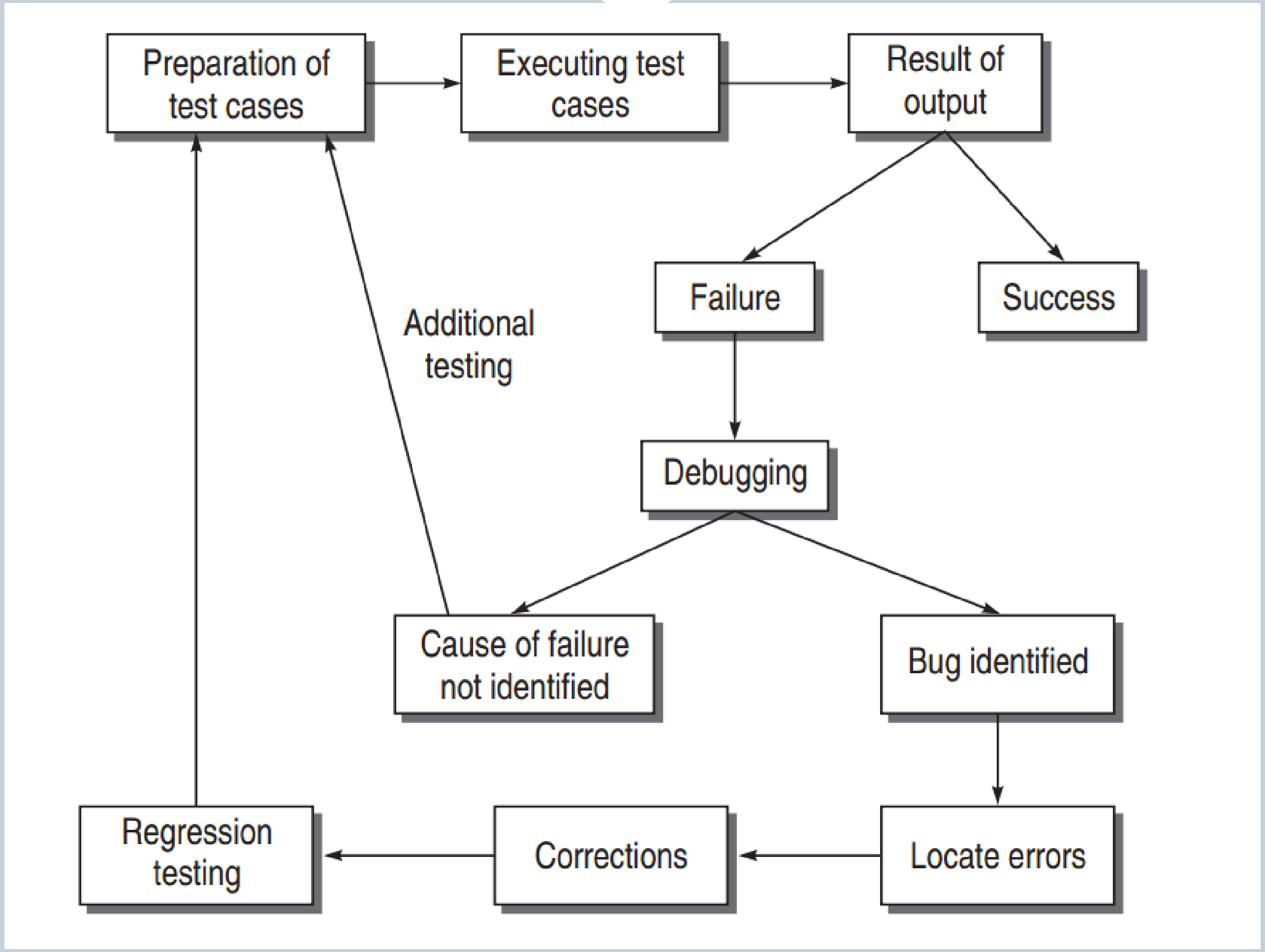
ABDUS SATTER
LECTURER
INSTITUTE OF INFORMATION TECHNOLOGY
UNIVERSITY OF DHAKA

Debugging

Debugging

- The goal of debugging process is to determine the exact nature of failure with the help of symptoms identified, locate the bugs and errors, and finally correct it

Debugging Process



Debugging is Difficult

- Debugging is performed under a tremendous amount of pressure which may cause problems instead of leading to clues of the problem. This pressure happens due to the following reasons:
  - Self-induced pressure to fix the suspected bug as early as possible since it is related to the individual performance and ego.
  - Organizational time-bound pressure to fix the bugs is always there.

Debugging is Difficult

- The gap between the faults and failures is very large in case of debugging of software systems. Without examining the whole program, we cannot locate the error directly. That is why, debugging starts with observing the failures for finding clues of the problem. With the clues, we then trace the bug and error. Moreover, the symptoms are also not clear after observing the failures. This makes debugging a time-consuming process.

Debugging is Difficult

- The complex design of the software affects the debugging process. Highly coupled and low-cohesive module is difficult to debug.
- Experience matters in the debugging process. A new member of the team will find it difficult or cumbersome as compared to experienced members.
- Sometimes, failures don’t happen. This does not mean that there is no bug. What happens is that we are unable to reproduce that failure always. This type of bug consumes a lot of time.

Debugging Techniques

- Debugging With Memory Dump
- Debugging With Watch Points

Debugging With Memory Dump

- In this technique, a printout of all registers and relevant memory locations is obtained and studied.
- There is difficulty of establishing the correspondence between storage locations and the variables in one's source program.
- The massive amount of data with which one is faced, most of which is irrelevant.
- It is limited to static state of the program as it shows the state of the program at only one instant of time.

Debugging With Watch Points

- Output statements
  - It may require many changes in the code. These changes may mask an error or introduce new errors in the program.
  - After analysing the bug, we may forget to remove these added statements which may cause other failures or misinterpretations in the result.
- Breakpoint execution
  - There is no need to compile the program after inserting breakpoints, while this is necessary after inserting output statements.
  - Removing the breakpoints after their requirement is easy as compared to removing all inserted output statements in the program.
  - The status of program variable or a particular condition can be seen after the execution of a breakpoint as the execution temporarily stops after the breakpoint. On the other hand in case of output statements, the full program is executed and output is viewed after the total execution of the program.

Categories of Breakpoints

- Unconditional breakpoint: It is a simple breakpoint without any condition to be evaluated. It is simply inserted at a watch point and its execution stops the execution of the program.
- Conditional breakpoint: On the activation of this breakpoint, one expression is evaluated for its Boolean value. If true, the breakpoint will cause a stop; otherwise, execution will continue.

Categories of Breakpoints

- Temporary breakpoint: This breakpoint is used only once in the program. When it is set, the program starts running, and once it stops, the temporary breakpoint is removed. The temporary nature is one of its attributes. In other respects, it is just like other breakpoints.
- Internal breakpoint: These are invisible to the user but are key to debugger's correct handling of its algorithms. These are the breakpoints set by the debugger itself for its own purposes.

Breakpoint

- Single-stepping
  - Step into
  - Step over

Correcting The Bugs

- Evaluate the coupling of the logic and data structure where corrections are to be made. Highly coupled module correction can introduce many other bugs. That is why low-coupled module is easy to debug.
- After recognizing the influence of corrections on other modules or parts, plan the regression test cases to perform regression testing as discussed earlier.
- Perform regression testing with every correction in the software to ensure that the corrections have not introduced bugs in other parts of the software.

Debugging Guidelines

- Fresh thinking leads to good debugging
- Don’t isolate the bug from your colleagues
- Don’t attempt code modifications in the first attempt
- Additional test cases are a must if you don’t get the symptom or clues to solve the problem
- Regression testing is a must after debugging
- Design should be referred before fixing the error

Debuggers

- **Kernel debugger:** It is for dealing with problems with an OS kernel on its own or for interactions between heavily OS-dependent application and the OS.
- **Basic machine-level debugger:** It is used for debugging the actual running code as they are processed by the CPU.
- **In-circuit emulator:** It emulates the system services so that all interactions between an application and the system can be monitored and traced. In circuit emulators sit between the OS and the bare hardware and can watch and monitor all processes and all interactions between applications and OS.
- **Interpretive programming environment debugger:** Debugger is well integrated into the runtime interpreter and has very tight control over the running application.

Thank You

END OF CHAPTER