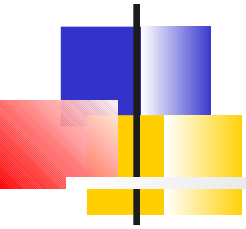
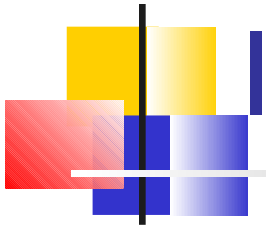


Debugging

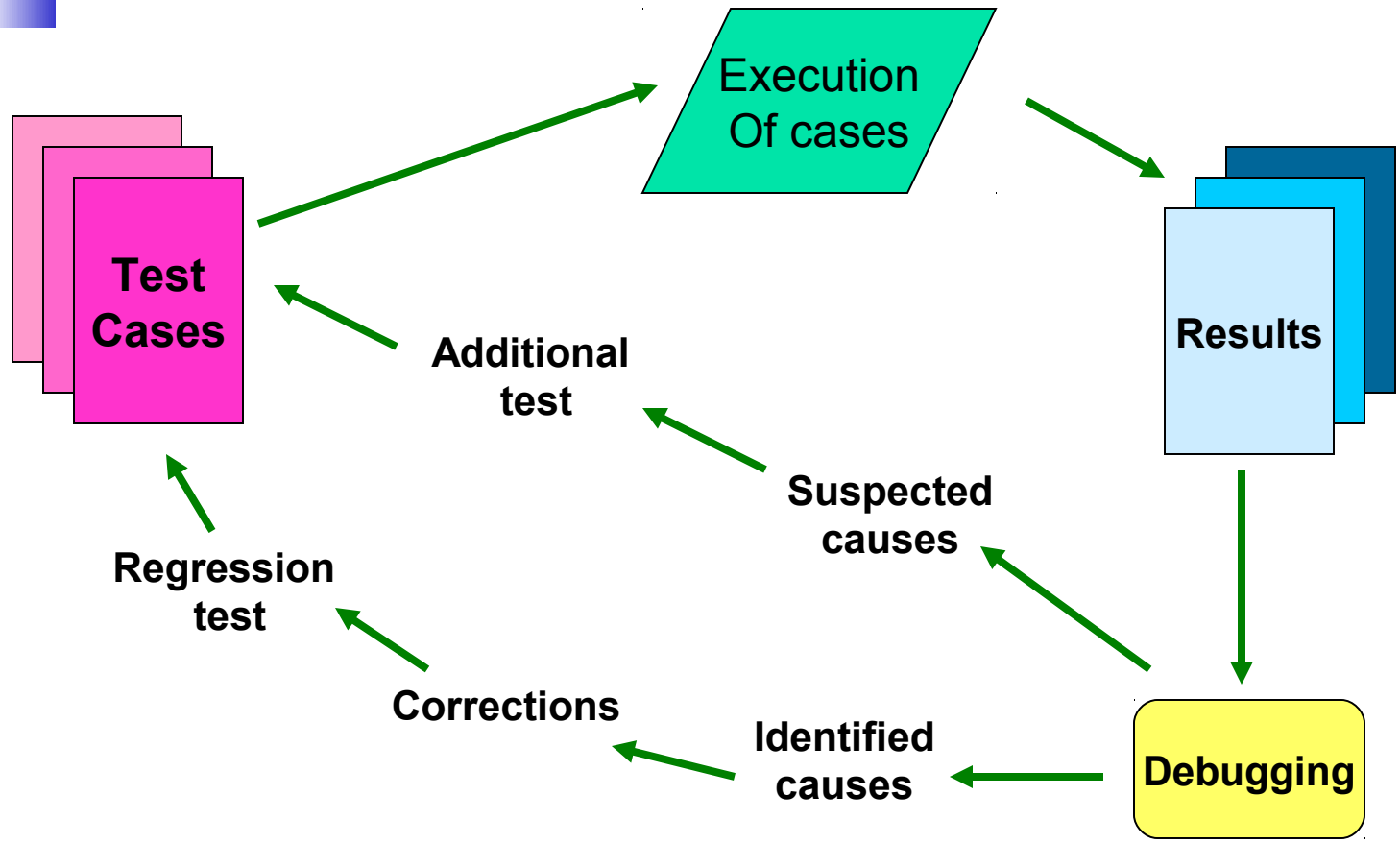
Indu Sharma
HOD(CSE)
CPTC, Rajsamand

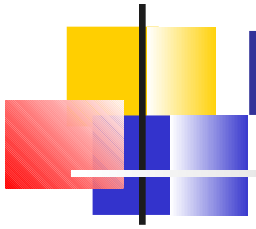


Introduction

- 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 has the objective: to find and correct the cause of a software error.

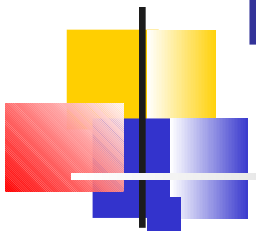
Debugging Process





Debugging Process

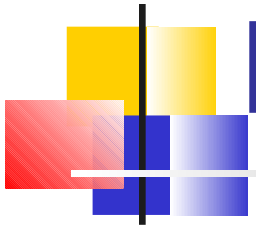
- The debugging process begins with the execution of a test case.
- Results are assessed and a lack of correspondence between expected and actual performance is encountered.
- The debugging process will always have one of two outcomes:
  1. The cause will be found and corrected, or
  2. The cause will not be found.
- In the latter case, the person performing debugging may suspect a cause, design a test case to help validate that suspicion, and work toward error correction in an iterative fashion.



Debugging Approaches

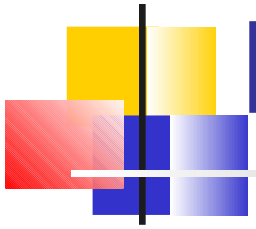
In general, there are three categories for debugging approaches:

1. Brute Force
2. Backtracking
3. Cause Elimination



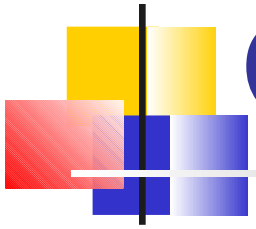
Brute Force

- Take memory dumps, invoke run time traces, program is loaded with PRINT statements.
- We hope that in the mass of information that is produced we will find a clue that can lead us to the cause of an error.
- Least efficient and quite common.
- More frequently leads to wasted effort and time.



Backtracking

- Can be used successful in small programs.
- Beginning at the site where a symptom has been uncovered the source code is traced backward (manually) until the site of the cause is found.
- Disadvantage: as the number of source line increases, the number of potential paths may become unmanageably large.



Cause Elimination

- Introduces the concept of binary partitioning.
- Uses binary partitioning to reduce the number of locations potential where errors can exist
- Isolate potential causes, devise cause hypotheses tests to isolate bug.