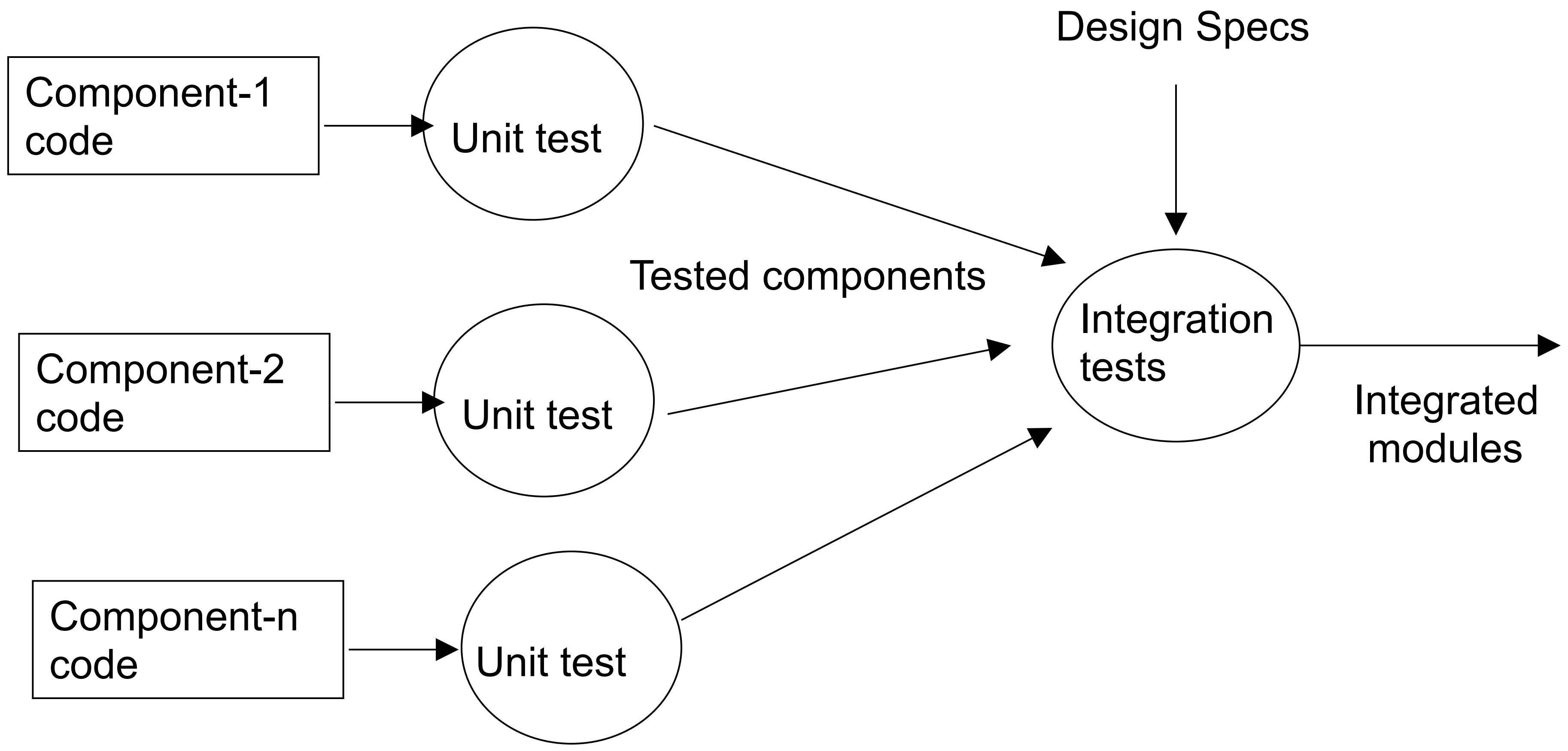
TESTING-2

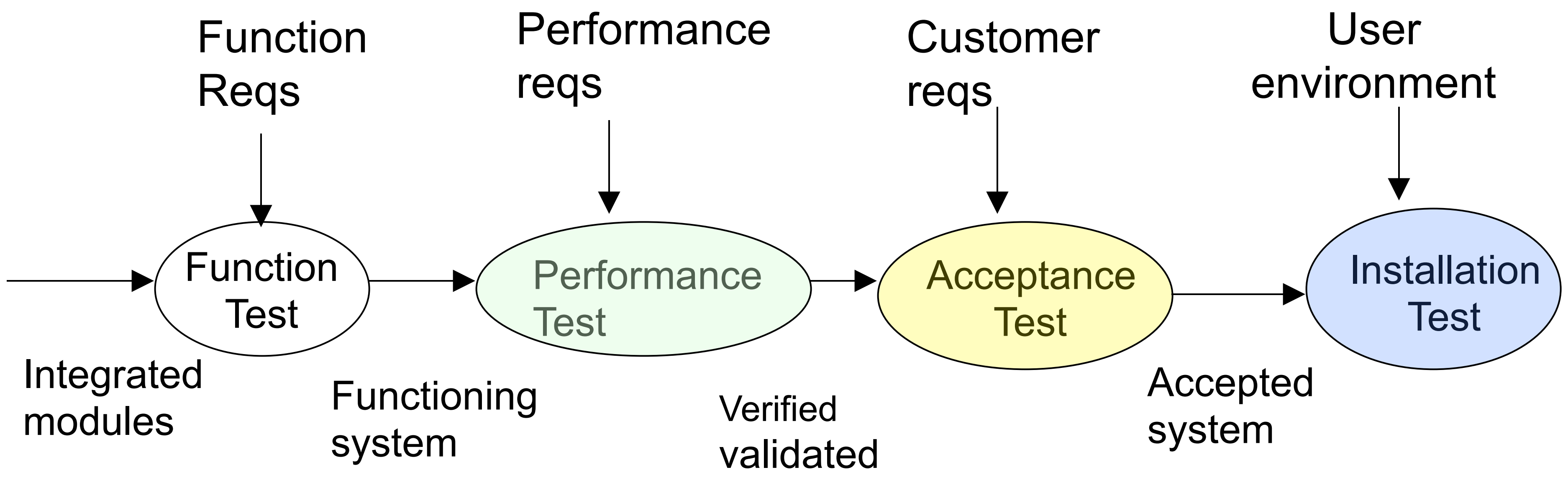
TESTING PROCESS

TESTING ISSUES

The testing process



The Testing process Contd



Unit Testing

Integration Testing

- Assemble tested components to form the subsystem
- Easier to integrate small pieces and test them - than to integrate the entire system and then test the whole system
  - Top-down, bottom-up etc strategies

Functional Testing

- Test all the functionality as per requirements
- Example: Word processing

Document Modification -	major functional group
add a char, word, para	
delete a char, word, para	
change format, font	
…..	

Performance Testing

- Load tests - load system using many users, devices etc
- Stress tests – over load system using many users, devices etc – see how it fails
- Recovery tests - response to faults and loss of data
- volume tests - test ability to handle large amounts of data
- configuration tests - test s/w and h/w configs
- compatability tests - test interfacing with other systems
- security tests
- reliability tests - up-time (Mean Time To Failure)
- Usability tests - test user interfaces
- and so on….

Acceptance Testing

- Benchmark tests etc
- Alpha test - pilot test run in-house
- Beta test - pilot test run at customer site
- Parallel testing - both existing and new system run in parallel (allows time to build up confidence in new system)

Installation Testing

- Usually involves running tests at customer site to verify working of installed system

Installation Testing

- Usually involves running tests at customer site to verify working of installed system

ISSUES IN TESTING

Two Major Issues in Testing

1. Testing is not same as proving! Goal is to find bugs in a smart way.

2. Testing is expensive, effort must be managed.

1) Testing is not same as proving!

Exhaustive Testing is impossible

- Black box
  - Number of test cases/scenarios too large
- White box
  - Number of paths too large
- From infinity – if you take some numbers the remaining is still infinity! Testing can't show bugs do not exist.
- More tests do not mean better testing. Lesser tests can do a better job! ???
  - 100000 tests vs 10 tests (which is better?)

2) Testing is expensive

- How much % of overall software development is devoted to testing?

- Why is it expensive?
  - ?
  - ?
  - ?

2) effort must be managed.

- Testing is an umbrella activity – can start once you start creating specifications.

- Testing involves a LOT of work/ It involves
  - specifying test cases, designing tests, creating tests, testing tests, adding/removing/changing tests, rerunning tests, reporting, tracking the effort etc
  - Risk based exercise: Cost of testing vs number of bugs missed.
- Also – defect tracking/management
- All this effort must be PLANNED out and tracked.

Other issues

- How to generate/select testcases? (bbox/ wbox)
- Judging Test Effectiveness: How good are the tests? (mutation testing)
- How to automate? What to automate?

A few Testing tools

- Here are a few tools
  - Unit testing: jUnit, Parameterized testing
    - djUnit (coverage)
  - Integration Testing: Mockito (stubbing)
  - System/Acceptance testing: GUI testing
  - Performance Testing: JMeter

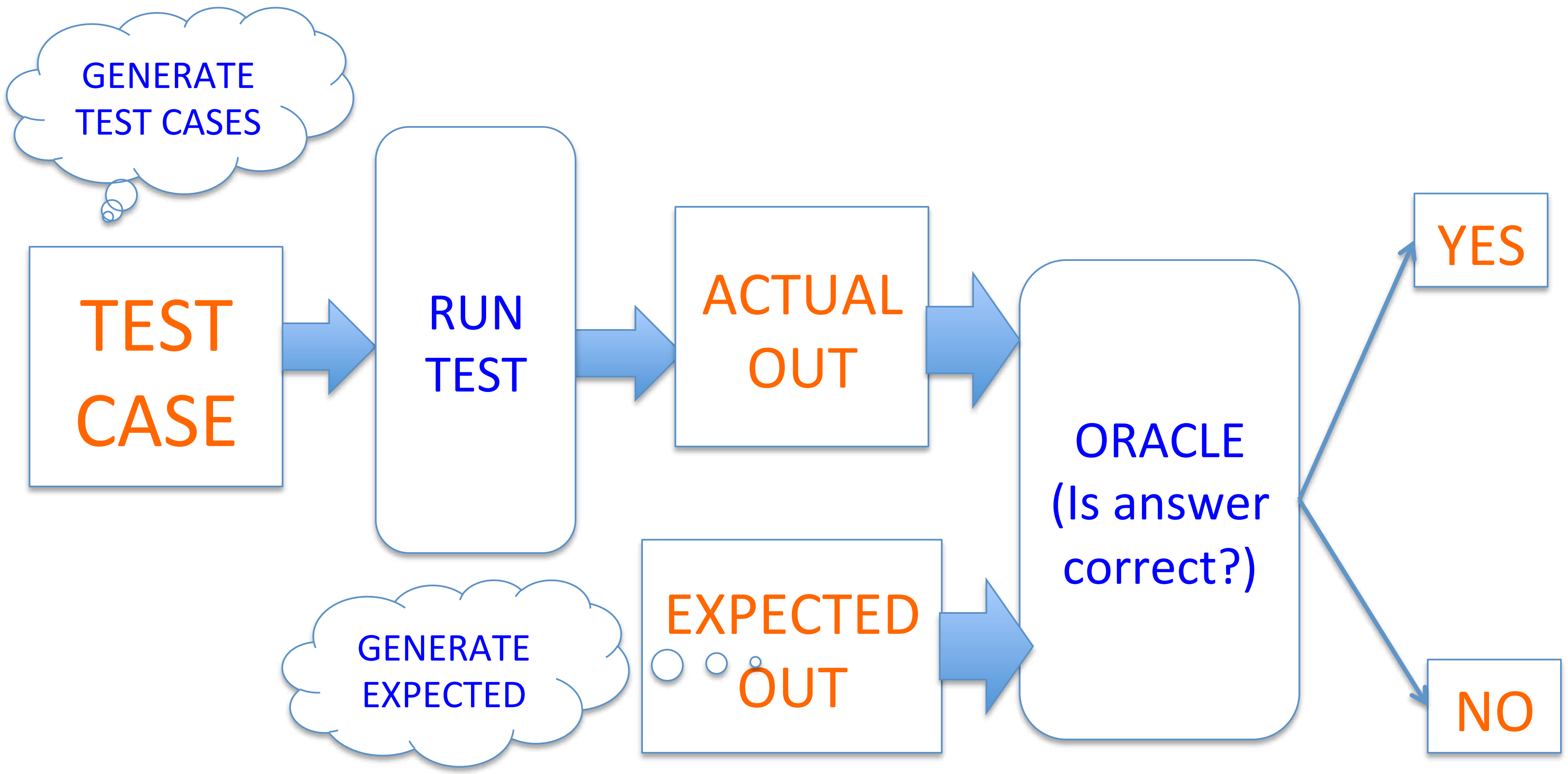
  - Stan4j, Metrics (Static analysis)
  - TPTP (execution profiling)
  - Bugzilla (Bug database/tracking)
- Automation is very important in testing.
  - Why?

Why automate?

- Manual testing is
  - ERROR PRONE
  - EXPENSIVE

- REGRESSION TESTING
  - Regress (go back)
  - Test again after changes have been made to software to check that the software does not regress (and break things that used to work).
  - Means re-testing after changes.

What to automate?



How to automate?

- automated test case generation?
- test drivers (like junit)
- test oracles
  - does away with expected output
  - checks if results are correct
  - example: check if results are sorted (how?)