Wk-9

L1 Testing - Motivation

- errors in software prog. → can cause entire sys. to crash Eg. Airport crash, etc.
- basic activities → provide input, check output, see if it matches exp. output, locate error, fix, test ∂.
- Test case [I → input, S → state, R → exp. result]
- Test suite → set of all Test cases which have been designed to test a given prog.
- testing for a large connec^n of randomly selected test cases do not guarantee that all errors will be uncovered.
- domain of all input values is too large, necessary to design minimal test suite where each test case helps detect diff. types of errors.
- blackbox → w/o looking into the code struc.
- whitbox → based on struc. of prog., test case.
- unit → indi. func^{ns} of prog. are tested.
- integra^n → units are incrementally integrated & tested after each step.
- sys. → fully integrated sys. is tested
  - → α → test team within org.
  - → β → selected customers
- acceptance → Customer determines whether or not to accept the software delivery.

L2 Unit Testing

- while testing a module, other modules with which this modules needs to interface may not be ready, unit testing makes debugging easier.
- when - during coding, who - dev., what - design unit test cases, build testing unit.

Drivers → contains non-local data struc. accessed by module under test.

↑
Module
↓
Stubs → dummy procedure with a highly simplified behaviour.

- Python tools — unittest, pytest.

L3 Black box & Whitbox Testing

- Black box → design test cases based on I/O values. No knowledge of design / code is req.
- Equivalence classes → domain of input values partitioned into a set of equivalence classes. Prog. behaves similarly for every input data in a particular equiv. class.
- BVA → examine the values at boundaries of the equivalence classes.
- Cov. Testing → branch, multiple condition, path.
- Branch cov. → every branch of the prog. needs to be taken atleast once.
- Multiple Condi^n → composite conditional exp. → each comp. condi^n takes T / F values.
- Path Cov. → A test suite achieves path cov., if it access all linearly indep. at least once.

Take help of CFGs.
→ CC ⇒ no. of decision / loop statement + 1.

L4 Integra^n & Sys. Testing

- Integra^n → starts when atleast few modules have undergone unit testing, obj. is to detect errors at the module interface.
- Big-Bang → all modules integrated in 1 step. Cons → difficult to find errors.

- Bottom-up - modules of each sub-sys. are integrated
  ↳ denver req.
- top-down - starts with root module. Req. stubs.
- Mixed - use both bottom-up & top-down app
- Smoke Testing → carried before initial sys. Testing. Check whether basic func^n are working.
- Performance Testing → Check whether sys. meets non-func^{nal} req.
- Stress Testing → input data vol., data rate, processing time, memory utilisa^n are tested beyond the designed capacity.

L5 TDD (Test Driven Dev.)

- used esp. in agile processes, writes tests first for the func., derive design & dev. based on tests.
- FAIL, PASS, & finally, refactor.