

### Testing Techniques

#### • Purpose of testing

- Verification - doing things right?
- Validation - doing right things?



#### • Types of tests

- White box
- Black box

Testing cannot prove absence of errors

Testing can only demonstrate presence of errors

---

---

---

---

---

---

---

---

### White Box Testing

#### • Internal details known

- Tests designed by studying code



#### • Goals

- Test all execution paths
- Test all boundary conditions
  - Min, max, middle, out-of-range

---

---

---

---

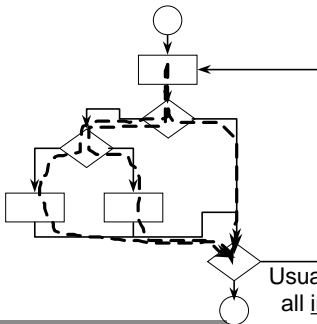
---

---

---

---

### White Box Example



Total paths =  $3^{20} = 3,486,784,401$

Loop up to 20 times

- Path 1
- Path 2
- Path 3

Usually settle for testing all independent paths

---

---

---

---

---

---

---

---

Testing Loops

n = maximum allowable passes

- Skip the loop entirely
- One pass through the loop
- Two passes through the loop
- m passes through loop ( $m < n$ )
- n-1, n, n+1 (?) passes

---

---

---

---

---

---

---

---

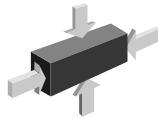
---

---

---

Black Box Testing

- Test against specification
- No knowledge of internals
- Inputs
  - In the valid range
  - At boundaries of valid range
  - Outside the valid range




---

---

---

---

---

---

---

---

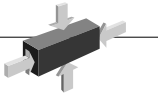
---

---

---

Black Box Testing

- Outputs
  - Apply inputs that produce boundary values at the output
- Comparison testing
  - Check against another implementation
    - Perhaps the system being replaced




---

---

---

---

---

---

---

---

---

---

---

Error Testing

- **Try to provoke all errors**
  - Invalid input, missing input, etc.
- **Evaluate error handling**
  - Adequacy of error reporting
  - Effectiveness of error recovery
- **Consider error propagation**

---

---

---

---

---

---

---

---

Test Strategy

- **Unit testing**
  - Drivers, stubs
- **Integration testing**
  - Merging of program units
- **Acceptance testing**
  - Performed by client/user




---

---

---

---

---

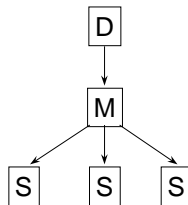
---

---

---

Unit Testing

- **One unit (module?) at a time**
- **Scaffolding**
  - Driver - replaces higher-level modules
  - Stubs - replace lower-level modules




---

---

---

---

---

---

---

---

Integration Testing

- **Direction of approach**
  - Top-down integration
  - Bottom-up integration
- **Style of integration**
  - Incremental
  - Big bang (!)

---

---

---

---

---

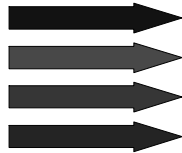
---

---

---

Iteration (?)

- **Simple model**
  - Build units
  - Test units
  - Integrate units
  - Test system
  - No iteration; one-pass process




---

---

---

---

---

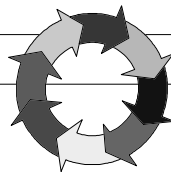
---

---

---

Iteration

- **Common practice**
  - Iterative development
  - Each unit adds features (and fixes!)
- **Build frequency**
  - Periodic - delayed feedback danger
  - Daily - reduces delay (but risky?)




---

---

---

---

---

---

---

---

Daily Build

- **Developers check in code**
  - Per QA release procedure
- **System build**
  - Often overnight and automated
- **Build validation**
  - Sanity testing (early morning?)

---

---

---

---

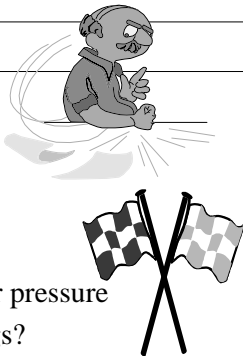
---

---

---

---

Daily Build



- **Management**
  - “Don’t break the build!”
- **Communication**
  - Public status - peer pressure
  - Red and green flags?

---

---

---

---

---

---

---

---

Regression Testing

- **Prevent “regression”**
  - Reappearance of previous defects
- **When defect found**
  - Create test that would exhibit error
  - Verify test, fix defect, test again
  - Add test to suite; execute on changes

---

---

---

---

---

---

---

---

Acceptance Testing

- **User/client/customer testing**
  - Primarily validation
- **Alpha test**
  - Customer at developer site?
- **Beta test**
  - At customer site (“beta site”)
  - Term “beta” is often misused today

---

---

---

---

---

---

---

---

Debug versus Test

- **Debug**
  - Driven by apparent defects
  - Does involve test construction
  - Ends when no defects are apparent
- **Test**
  - Most useful when no apparent defects




---

---

---

---

---

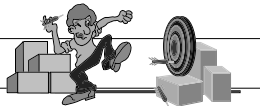
---

---

---

Choosing Tests

- **Limited testing resources**
  - Money, time, people
- **What tests to do?**
  - Path coverage (white box)?
  - Validation (black box)?
  - User scenarios?
    - Statistically weighted?




---

---

---

---

---

---

---

---

Who Does Testing?

- **Success == find defects (?)**
- **Software developer**
  - Pride in work
  - Embedded in own model of system
- **Tester**
  - Independent “outsider” [not adversary]

---

---

---

---

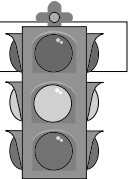
---

---

---

---

End of Testing?



- **When time runs out?**
- **When money runs out?**
- **When enthusiasm wanes?**
- **Some models exist**
  - Predict remaining defects
  - Economic analysis

---

---

---

---

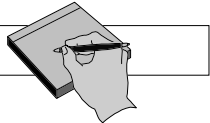
---

---

---

---

Test Plan



- **Written test sequence**
- **User input**
  - Script detailing sequence of operations
- **File or database input**
  - Sample files and expected output

---

---

---

---

---

---

---

---

Configuration Management

- **Manage all work products**
  - Code
  - Documentation
  - Baseline releases
  - Test plans
- **Control updates**

---

---

---

---

---

---

---

---

Source Control Tools

- **Low-level CASE tools**
- **Store work products**
  - Program source code
  - Analysis/design documents & models
- **Maintain all past history**
  - Can recreate prior versions

---

---

---

---

---

---

---

---

Example Tools

- **Microsoft Windows**
  - Visual SourceSafe
  - MKS Source Integrity
  - PVCS
- **UNIX**
  - SCCS
  - RCS, CVS

"Delta" format stores one version and changes needed to reconstruct other versions.

---

---

---

---

---

---

---

---

...

Revision Control Procedure

- Check out module from system
- Make changes
- Check module back in
- Create new project version
  - Optional

“Check-in” typically controlled by SCM group.

---

---

---

---

---

---

---

---