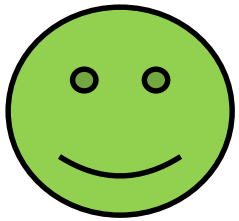
Unit-4 ()
(Lecture Notes)

Prepared by
Jay Nanavati, Assistant Professor, SEMCOM

Topics

- Coding - Introduction
- Top-down and Bottom-up Coding
- Structured Programming, Information Hiding
- Programming Style
- Testing
- Levels of Testing
- Functional Testing vs. Structural Testing

Coding - Introduction

- What is coding?
- Goal of coding
- Coding-Testing-Maintenance
- Easy-to-write programs
- Easy-to-read and Easy-to-understand programs
- Input: _____ Output: _____

Top-down vs. Bottom-up Coding

- All designs contain hierarchy.
- Hierarchy of functional modules vs. objects.
- In top-down approach, the implementation starts from the top of hierarchy and proceeds to lower levels. First main module is implemented then its subordinates are implemented and their subordinates and so on.
- In Bottom-Up approach the process is reverse. Here bottom level modules are implemented first and this proceeds through the higher levels until it reaches the top.

Top-down vs. Bottom-up Coding (contd.)

- When we proceed top-down for testing a set of modules at the top of the hierarchy, stubs will have to be written for the lower-level modules that the modules under testing invoke.
- When we proceed bottom-up, all modules that are lower in hierarchy have been developed and driver modules are needed to invoke these modules under testing.
- Which approach should be used?

Structured Programming

- The structured programming movement started in 1970s.
- Structured Programming is often regarded as ‘goto-less’ programming.
- A program has static structure as well as dynamic structure.
- It will be easier to understand the dynamic structure of program if it resembles the static structures.

Structured Programming (contd.)

- The goal of structured programming is to ensure that the static structure and dynamic structures are same.
- No meaningful program can be written as sequence of simple statements without any branching or repetition.
- A statement in structured programming is structured statement - that it has single-entry and a single-exit.
- Structured statements lead to a structured program.

Information Hiding

- A software solution to a problem contains data structures which store only relevant information.
- In the problem domain, only certain operations are performed on information.
- **When the information is represented as data structures, only some operations should be performed on the data structures.**

Information Hiding (contd.)

- It means the information captured in data structures should be hidden from the rest of system.
- Information Hiding can reduce coupling.
- Another form information hiding is to let a module see only those items needed by it and other data items are hidden from such modules.
- Many modern programming languages support information hiding in form of data abstraction.
- In data abstraction, a module or package is defined that encapsulates the data along with some operations defined to be performed on to encapsulated data.

Programming Style

- What to do and what not to do (to produce simple readable code) ?

Names	User-defined types	Program Layout
Control Constructs	Nesting	Side effects
Gotos	Module size	Robustness
Information Hiding	Module Interface	

Programming Style (contd.)

- Name (of variables/methods)
  - ✓ Names should be closely related to the entity they represent.
  - ✓ Module names should reflect their activity.
  - ✓ Avoid using same names for different attributes/entities.

Rate of interest	interest_rate	roi , ir etc.
Total marks	total_marks	tm, ttl_mrks etc.
Firstname	firstname	fname, fnm etc.
Average	average	a, avg etc.

Programming Style (contd.)

- Control constructs
  - ✓ Single-entry, single-exit constructs should be used.
  - ✓ A few standard constructs should be used, not all.
  - ✓ Be consistent.
- Gotos
  - ✓ Avoid using gotos.
  - ✓ Use forward jumps.

Programming Style (contd.)

- Information Hiding
  - ✓ Should be used wherever it is possible.
- User-defined Types
  - ✓ Exploit such facility offered by a programming language.
  - ✓ e.g. enum for weekdays, months, years etc.
- Nesting
  - ✓ Avoid (deep) nesting.
  - ✓ Use if-else if-else ladder with piped conditions.

Programming Style (contd.)

- Module Size
  - ✓ No hard-and-fast rules can be there.
  - ✓ Coupling and cohesion should be considered.
- Module Interface
  - ✓ Interface of a module should not be complex.
  - ✓ Upto five parameters is acceptable.
- Program Layout
  - ✓ Proper indentation, white spaces, parenthesis should be used.

Programming Style (contd.)

- Side Effects
  - ✓ Modules should be examined for possible side-effects.
  - ✓ Side effects must be documented.
- Robustness
  - ✓ Robustness is the ability to tolerate faults/exceptional conditions in graceful manner.
  - ✓ Validate inputs, provide exception handling code.

Internal Documentation

- Done with the help of comments.
- All programming languages support comments.
- Users do(or can) not read the comments.
- Comment on what the code is doing & not how/why the code is doing it.
- Use comments for code which is difficult to understand.

Internal Documentation (contd.)

- Provide comments for modules. (*prologue*)
- A prologue should describe:
  - ✓ functionality of module
  - ✓ parameters & their purpose
  - ✓ assumptions about inputs, if any.
  - ✓ global variables.
- Other information can also be included.
- Prologue should be up-to-date.

Code Verification

- Aims at showing that the code is consistent with the design it is supposed to implement.
- Two categories: static and dynamic methods.
- Static methods do not involve actual program execution.
- Static methods include program verification, code reading, code reviews & walkthroughs etc.
- In Dynamic methods, the program is executed on test data & the outputs are examined.
- Dynamic methods include testing.

Code Reading

- What is code reading?
- Who does it?
- What is the goal?
- How is it done?

Software Testing

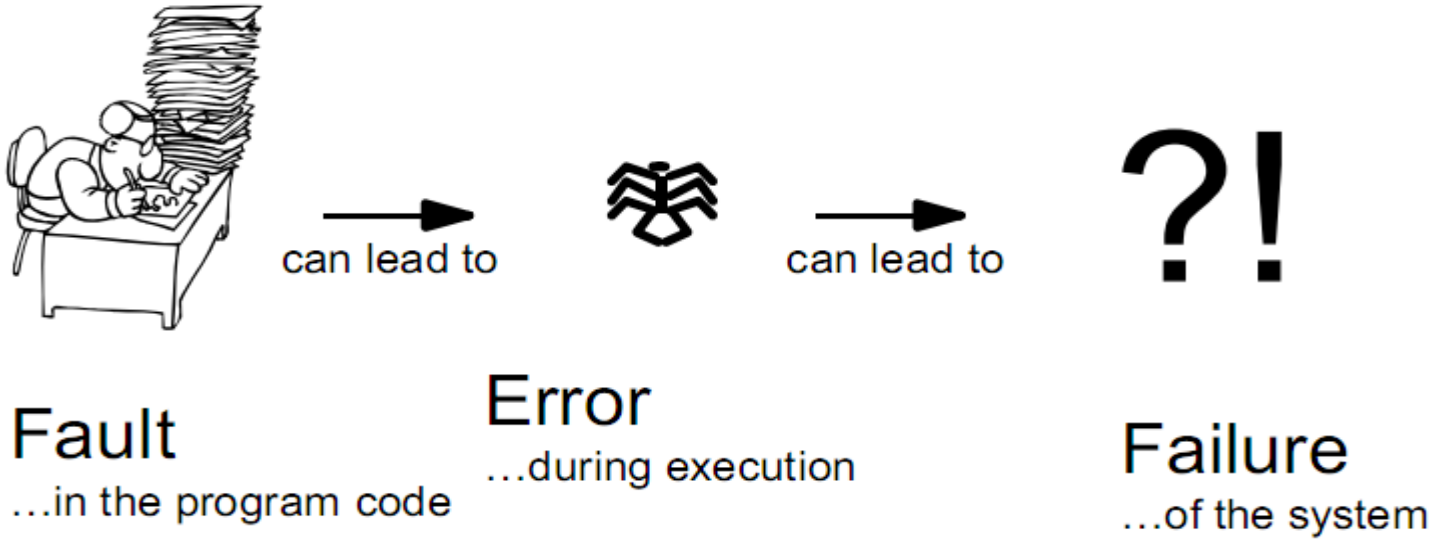
- Error can be injected at any stage during software development.
- Verification at the end of every stage is just not enough.
- Testing tries to detect *all remaining* errors.
- Testing is the *last* chance to detect & to fix errors before the software is deployed.
- *Integration &* Testing
- Test cases – are critical.

Error, Fault, Failure & Reliability

- Error refers to the discrepancy (i.e. variance) between a computed, observed or measured value and the true, specified or theoretically correct value.
- Fault is a condition that causes a system to fail in performing its required function.
- A fault is the basic reason for software malfunction.

Error, Fault, Failure & Reliability (contd.)

- Failure is the inability of a system or component to perform a required function according to its specifications.
- Relationship between failure & faults



Error, Fault, Failure & Reliability (contd.)

Requirements:

Calculate the hypotenuse of a right-angled triangle.

I.e., for inputs a and b, produce output $\sqrt{a^2+b^2}$

Implementation:

```
a = readFloat();
b = readFloat();
tmp = pow(a,2) * pow(b,2);
c = sqrt(tmp);
printFloat(c);
```

fault

User interaction:

input: 3

intput: 4

failure

output: 12

Execution:

a = 3

b = 4

error

tmp = 144

error

c = 12

Error Propagation

Example Courtesy : www.mdh.se/idt

Top-down & Bottom-up Approaches

- A large & complex system consists of hierarchy of modules.
- So, it is not possible to test the entire system.
- Instead, Incremental testing is used.
- Two ways to combine modules:
  1. Top-down
  2. Bottom-up

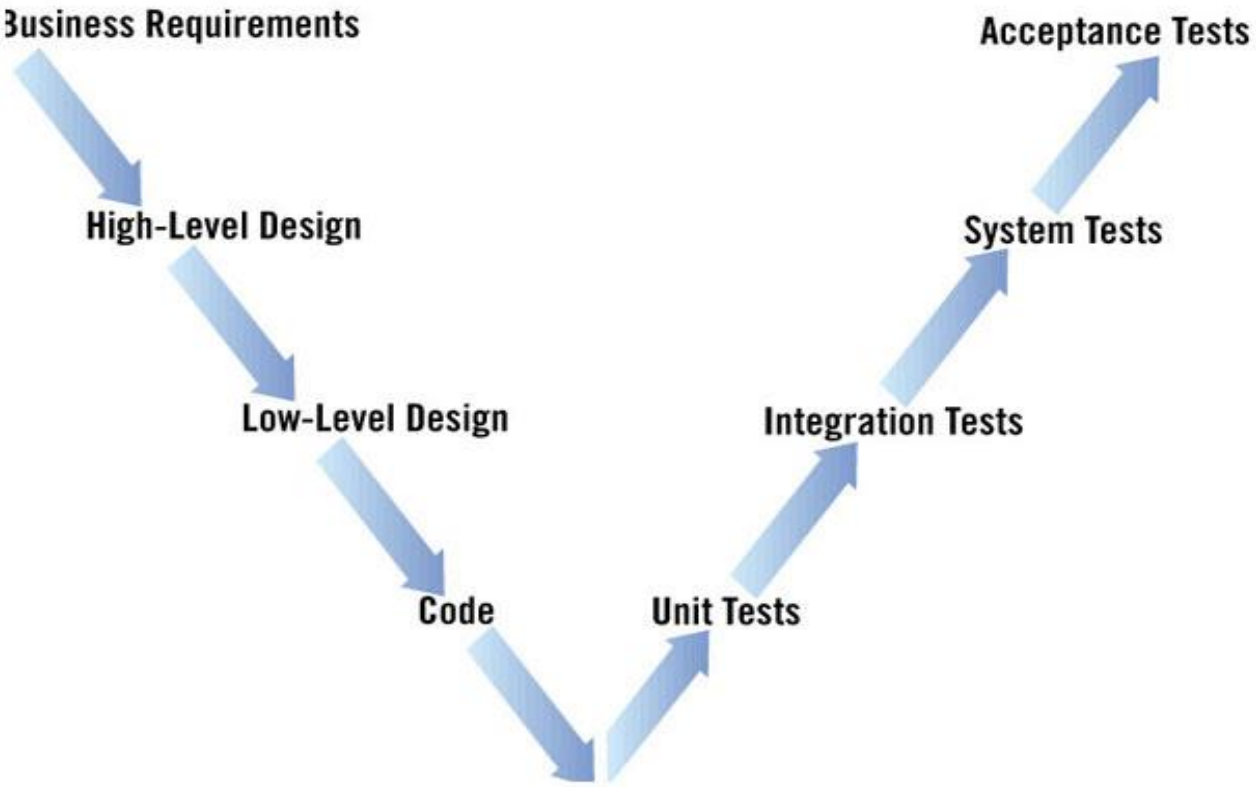
Top-down & Bottom-up Approaches (contd.)

- In top-down approach we start by testing the top of hierarchy and incrementally add modules that it calls and then test new combined system.
- This requires stub to be written. A stub is a dummy routine that simulates a module that is subordinate to the top-level module, which is not yet coded.
- In bottom-up approach we start testing from the bottom level modules, which are then combined with the higher-level modules.
- For this drivers are needed to set up appropriate environment and invoke the module.

Top-down & Bottom-up Approaches (contd.)

- Writing stubs is more difficult than writing drivers because writing stubs requires us to know beforehand the inputs of the modules being simulated.
- The testing method should be similar to that of the development method.
- That is if system is developed in top-down manner then top-down testing is used. If system is developed in bottom-up manner then bottom-up approach should be used.

Levels of Testing



Levels of Testing (contd.)

- Unit testing
  - ✓ The first level of testing is Unit Testing in which different modules are tested against specifications produced during design.
  - ✓ The goal of unit testing is to test internal logic of modules.
  - ✓ It is done by programmer. Structural Testing is best suited for this level.
- Integrations Testing
  - ✓ In Integration Testing many unit-tested modules are combined into subsystems, which are tested.
  - ✓ The goal here is to test if the modules can be integrated properly.
  - ✓ Here emphasis is on testing interfaces between the modules.

Levels of Testing (contd.)

- System testing
  - ✓ The entire software system is tested.
  - ✓ The reference document for this process is the requirements document.
  - ✓ The goal is to see if software meets the requirements.
- Acceptance Testing
  - ✓ Acceptance Testing is performed with realistic data of the client to demonstrate that the software is working satisfactorily.
  - ✓ It focuses on external behavior of the system.

Levels of Testing (contd.)

- Regression Testing
  - ✓ It is performed when some changes are made to an existing system.
  - ✓ As modifications have been to an existing system, testing has to be done to make sure that new features indeed work and has not had any side effects on older services.
  - ✓ For regression testing old test cases are again executed on the modified system and its output is compared with earlier output to make sure that system is working as before on these test cases.

Functional Testing vs. Structural Testing

	Functional Testing	Structural Testing
1.	The structure of program is not considered for testing.	The structure of program is considered for testing.
2.	It is concerned with testing functionality of program.	It is concerned with testing implementation of program.
3.	Test cases are decided on basis of specifications of program or module.	Test cases are decided on the basis of the internal logic of program or module.
4.	Criteria for testing is imprecise.	Criteria for testing is precise and formal.
5.	It is called Black box testing.	It is White Box Testing.

Black Box Testing

- It is also known as functional testing.
- Software to be tested is treated as a *block box.*
- Specification for the black box is given
- Premise: Expected behavior is specified.
- Hence just the specified expected behavior is tested.
- How it is implemented is not an issue.

Black Box Testing (contd.)

- The expected behavior of the system is used to design test cases i.e test cases are determined solely from specification.
- Internal structure of code is not used for test case design.

White Box Testing

- White box testing focuses on implementation.
- Aim is to exercise different program structures with the intent of uncovering errors.
- It is also called structural testing.
- Various criteria exist for test case design.
- Test cases have to be selected to satisfy coverage criteria.