System Analysis and Design

Eighth Edition

Alan Dennis, Barbara Wixom, Roberta M. Roth

Chapter 11

Moving into Implementation

Objectives

- Be familiar with the system construction process.
- Explain different types of tests and when to use them.
- Describe how to develop user documentation.

Introduction

- As the implementation phase begins, foremost on people’s minds is construction of the new system
- A major component of building the system is writing programs
- The implementation phase consists of developing and testing the system ’ s software, documentation, and new operating procedures
- During this phase, it is also the responsibility of the systems analysts to finalize the system documentation and develop the user documentation

Managing the Programming Process

- Assigning the programmers
- Coordinating the activities
- Managing the schedule

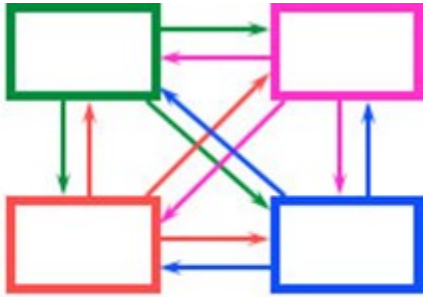
The Programmer Paradox

- More is not always better than less!
- After the “right” number of people are assigned to a programming task, adding more people slows down rather than speeds up completion of the project
- Projects requiring a large team should be broken into a series of independent, smaller parts

Assigning Programmers

- Minimize the number of programmers
- Match programming tasks with programmer capabilities
- When skills are deficient, apply mentoring and training

Two communication channels



Twelve communication channels

Coordinating Activities

- Weekly (hopefully brief) meetings
- Create and follow standards
- Organize programmers' work areas
  - Development area
  - Testing area
  - Production area
- Implement change control mechanisms
- Use program log to monitor program changes
- Many CASE tools are set up to track the status of programs and help manage programmers as they work

Managing the Schedule

- Use initial time estimates as a baseline
- Revise time estimates as construction proceeds
- Fight against scope creep
- Monitor “minor” slippage
- Create risk assessment and track changing risks
- Fight the temptation to lower quality to meet unreasonable schedule demands

Avoid Classic Mistakes

- Research-oriented development
  - If you use state-of-the art technology, lengthen planned time
- Using “low-cost” personnel
  - If using a significant number of entry level personnel, lengthen planned time
- Lack of code control
  - Use source code library to keep programmers from changing the same code at the same time
- Inadequate testing
  - Always allocate sufficient time for formal testing

Testing

- Writing programs is a fun, creative activity
- Testing and documentation are not fun
- Testing helps ensure that the system performs as outlined in the specifications
- It is dangerous to test early modules without an overall testing plan
- It may be difficult to reproduce sequence of events causing an error
- Testing must be done systematically and results documented carefully

Test Planning

- Testing starts with the tester's developing a ***test plan*** that defines a series of tests that will be conducted
- Today, automated testing tools enable comprehensive testing of all relevant test conditions

Test Plan Page ___ of ___

Program ID: _____ Version number: _____

Tester: _____ Date designed: _____ Date conducted: _____

Results: ☐ Passed ☐ Open items:

Test ID: _____ Requirement addressed: _____

Objective:

Test cases

Interface ID	Data Field	Value Entered
1.		
2.		
3.		
4.		
5.		
6.		

Script

Expected results/notes

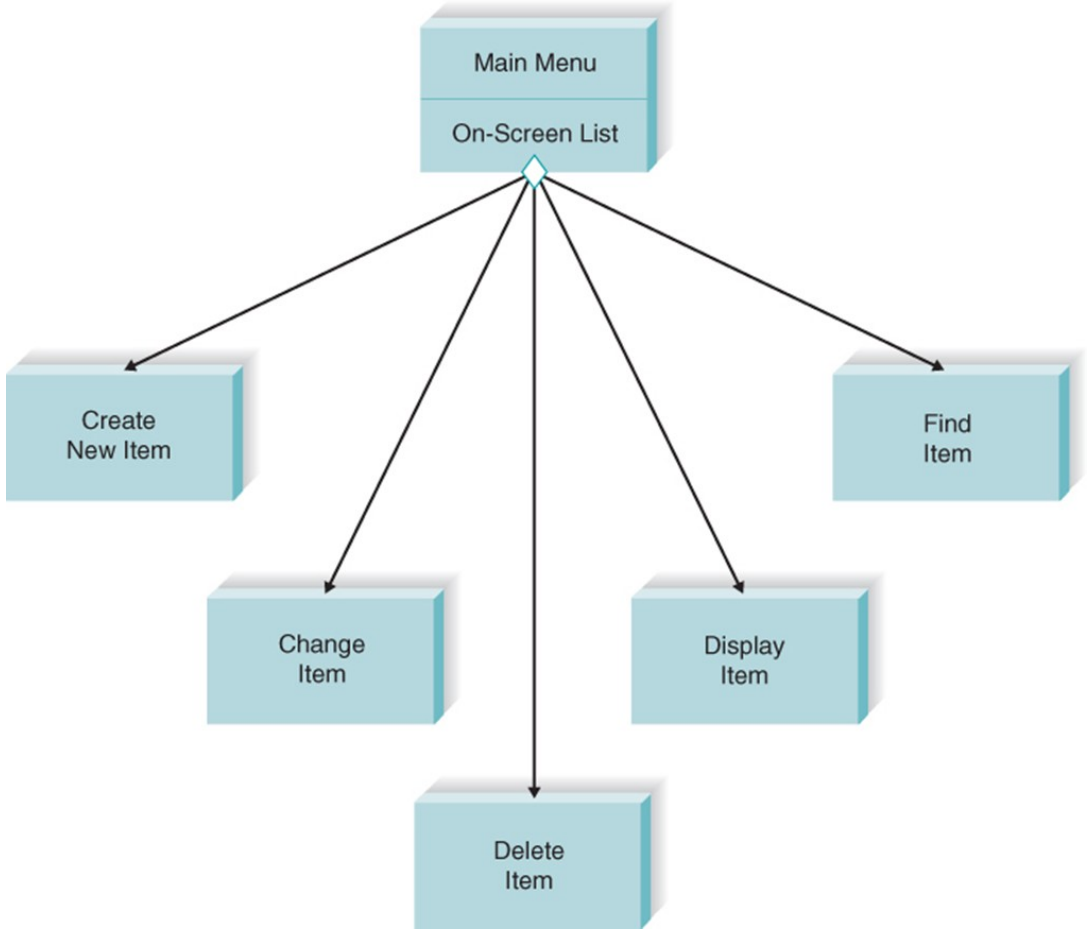
Actual results/notes

Categories of Testing

- Stub testing
  - Tests control structures before all modules are written
- Unit testing
  - Tests each module – Does it performs its function?
- Integration testing
  - Tests the interaction of modules - do they work together?
- System testing
  - Tests to assure that the software works well as part of the overall system
- Acceptance testing
  - Tests to assure that the system serves organizational needs

Stub Testing

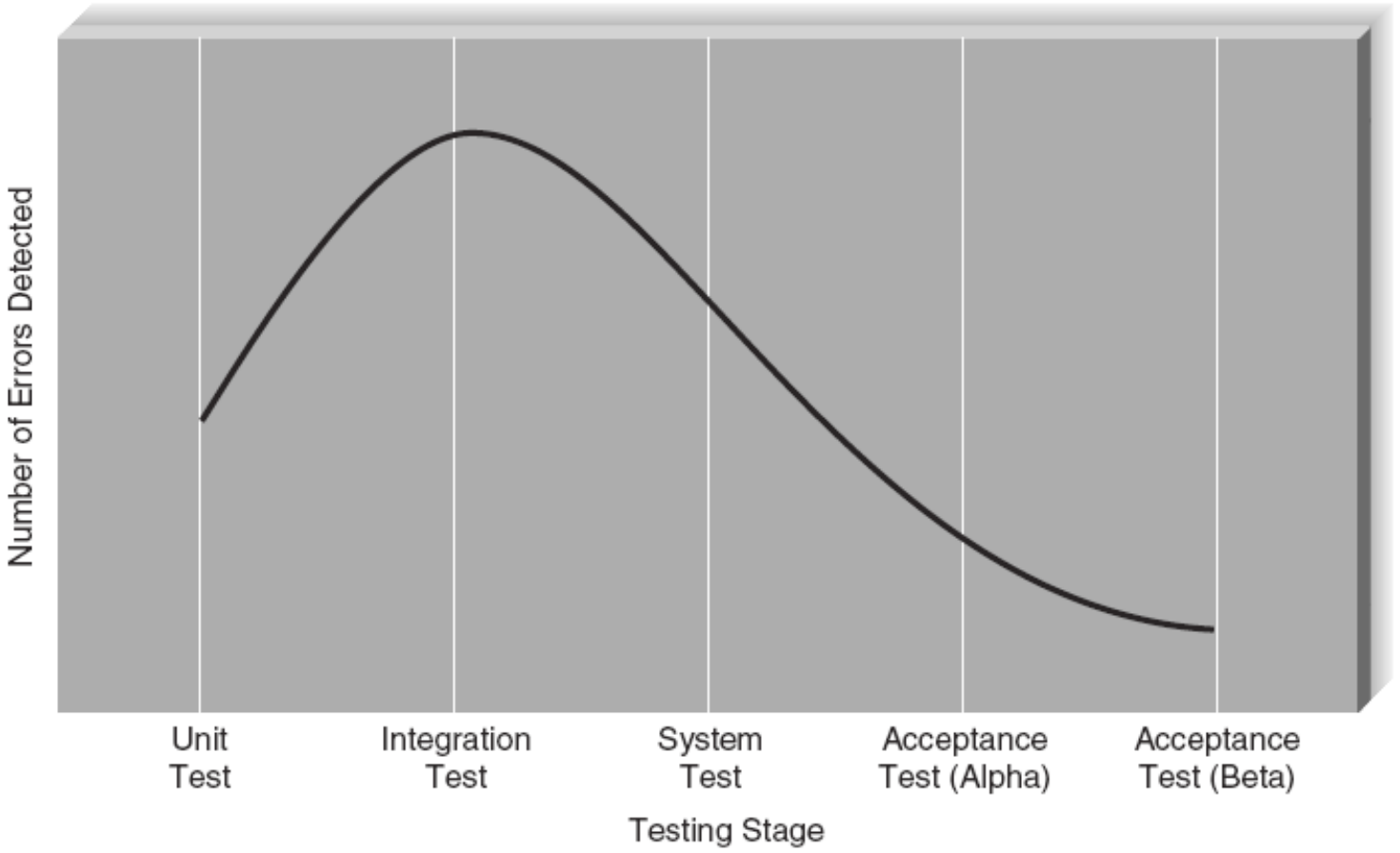
- Not all program modules are likely to be finished at the same time
- Programmer usually writes stubs for the unfinished modules to enable the modules around them to be tested
- A stub displays a simple test message on the screen or returns some hardcoded value



Four General States of Tests

1. Unit tests
2. Integration tests
3. System tests
4. Acceptance tests

Error Discovery Rates for Different Stages of Tests



Unit Testing

- ***Unit tests*** focus on one unit—a program or a program module that performs a specific function that can be tested
  - Performed after programmer has developed and tested the code and believes it to be error free
- Black Box Testing
  - Focuses on whether the unit meets requirements stated in specification
- White-Box Testing
  - Looks inside the module at actual code

Integration Tests

- User interface testing
  - Tests each interface function
- Use-scenario testing
  - Ensures that each use scenario works correctly
- Data flow testing
  - Tests each process in a step-by-step fashion
- System interface testing
  - Ensures data transfer between systems

System Tests

- Requirements testing
  - Ensures that integration did not cause new errors
- Usability testing
  - Tests how easy and error-free the system is in use
- Security testing
  - Assures that security functions are handled properly
- Performance testing
  - Assures that the system works under high volumes of activity (example: simultaneous users, peak transaction volume)
- Documentation testing
  - Analysts check the accuracy of documentation

Acceptance Tests

- Alpha testing
  - Performed by users to assure they accept the system; frequently repeats earlier tests
- Beta testing
  - Uses real data, not test data. Actual users monitor for errors or needed improvements.
- User sign-off following acceptance testing indicates the system is ready to be placed into production

Developing Documentation

- Documentation provides information to make the system easier to use and repair
- System documentation
  - Intended to help programmers and analysts understand and maintain the system after it is installed
- User documentation
  - Intended to help users operate the system

Producing Documentation

- High quality documentation takes about 3 hours per page or 2 hours per screen
- The task should not be left to the end of the project
- Time required to develop, and test user documentation should be built into project plan
- On-line documentation is predominant today

Value of Online Documentation

1. Searching is simplified
2. Information can be presented in multiple formats
3. New methods of interacting with documentation are possible (example: tool tips, animated demos, narrated demos)
4. Less costly than paper documentation

Types of User Documentation

Type of Documentation	Use
Reference documents	Designed to be used when the user needs to learn how to perform a specific function
Procedures manuals	Describe how to perform business tasks
Tutorials	Teach people how to use major components of the system

Guidelines for Crafting Documentation Topics

- Do not omit any step because you “assume” the user knows how to do that step
- Use the active voice with direct instructions
- Use consistent terms
- Use simple, friendly language
- Use parallel grammatical structure
- Use steps correctly (as actions)
- Use short paragraphs

Chapter Review

- Identify and describe the essential aspects of managing the programming process.
- Identify and describe the elements of a test plan.
- Identify and describe the focus and types of unit testing.
- Identify and describe the focus and types of integration testing.
- Identify and describe the focus and types of system testing.
- Identify and describe the focus and types of acceptance testing.
- Identify and describe the focus and types of user documentation.

Key Terms

- Acceptance tests
- Alpha testing
- Beta testing
- Black-box testing
- Change control
- Construction
- Data flow testing
- Documentation navigation control
- Documentation testing
- Documentation topic
- Hardcoded
- Integration tests
- Performance testing
- Procedures manuals
- Program log
- Reference documents
- Requirements testing
- Scope creep
- Security testing
- Stub
- System documentation
- System interface testing
- System tests
- Test case
- Test plan
- Tutorials
- Unit tests
- Usability testing
- User documentation
- User interface testing
- Use scenario testing